

Отчёта по лабораторной работе №8

Программирование цикла. Обработка аргументов командной строки.

сибомана Ламек НКАбд-03-24

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	7
4	Выводы	14

Список иллюстраций

3.1	Текст программы lab8-1.asm	7
3.2	Текст программы lab8-1.asm	8
3.3	Создание и запуск lab8-1.asm	8
3.4	Создание lab8-2.asm	8
3.5	Текст программы lab8-2.asm	8
3.6	Создание и запуск lab8-2.asm	9
3.7	Создание lab8-3.asm	9
3.8	Текст программы lab8-3.asm	10
3.9	Создание и запуск lab8-3.asm	10
3.10	Текст программы lab8-3.asm	11
3.11	Создание и запуск lab8-3.asm	11
3.12	Создание lab8-4.asm	12
3.13	Создание и запуск lab8-4.asm	13
3.14	Создание и запуск lab8-4.asm	13

Список таблиц

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. На рис. 8.1 показана схема организации стека в процессоре. Стек имеет вершину, адрес последнего добавленного элемента, который хранится в регистре esp (указатель стека). Противоположный конец стека называется дном. Значение, помещённое в стек последним, извлекается первым. При помещении значения в стек указатель стека уменьшается, а при извлечении — увеличивается. Для стека существуют две основные операции:

- добавление элемента в вершину стека (push);
- извлечение элемента из вершины стека (pop).

3 Выполнение лабораторной работы

1. Сначала я создал каталог для программ лабораторной работы № 8, затем перешёл в него и создал файл lab8-1.asm (рис. fig. ??) ![Создание каталога и файла lab8-1] (image/1.png){#fig:001 width=70%}

Открывал файл в Midnight Commander и заполняем его в соответствии с листингом 8.1 (рис. fig. ??).

![Текст программы lab8-1.asm] (image/2.png){#fig:002 width=70%}

Создал исполняемый файл и проверьте его работу.

![Создание и запуск lab8-1.asm] image/3.png){#fig:003 width=70%}

Изменил текст программы добавив изменение значение регистра esx в цикле. Снова открывал файл для редактирования и изменяем его, добавив изменение значения регистра в цикле (рис. fig. 3.1)

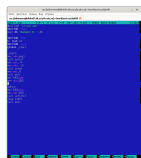


Рис. 3.1: Текст программы lab8-1.asm

Создаем исполняемый файл и запускаем его (рис. fig. ??).

![Создание и запуск lab8-1.asm] image/5.png){#fig:005 width=70%}

Регистр esx принимает значения 9,7,5,3,1 (на вход подается число 10, в цикле label данный регистр уменьшается на 2 командой sub и loop).

Число проходов цикла не соответствует числу N, так как уменьшается на 2.

Снова открываем файл для редактирования и изменяем его, чтобы все корректно работало (рис.fig. 3.2).

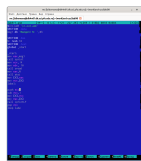


Рис. 3.2: Текст программы lab8-1.asm

Создаем исполняемый файл и запускаем его (рис. fig. 3.3).

```
lsibomana@dk4n61 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
lsibomana@dk4n61 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
lsibomana@dk4n61 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
Ошибка сегментирования (образ памяти сброшен на диск)
lsibomana@dk4n61 ~/work/arch-pc/lab08 $
```

Рис. 3.3: Создание и запуск lab8-1.asm

В данном случае число проходов цикла равна числу N.

ОБРАБОТКА АРГУМЕНТОВ КОМАНДНОЙ СТРОКИ

Создаем новый файл (рис.fig. 3.4).

```
lsibomana@dk4n61 ~/work/arch-pc/lab08 $ touch lab8-2.asm
lsibomana@dk4n61 ~/work/arch-pc/lab08 $
```

Рис. 3.4: Создание lab8-2.asm

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 8.2 (рис.fig. 3.5).

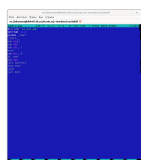


Рис. 3.5: Текст программы lab8-2.asm

Создаем исполняемый файл и проверяем его работу, указав аргументы (рис. fig. ??).

```
lsibomana@dk4n61 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
lsibomana@dk4n61 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
lsibomana@dk4n61 ~/work/arch-pc/lab08 $ ./lab8-2 1 2 3
1
2
3
lsibomana@dk4n61 ~/work/arch-pc/lab08 $
```

Рис. 3.6: Создание и запуск lab8-2.asm

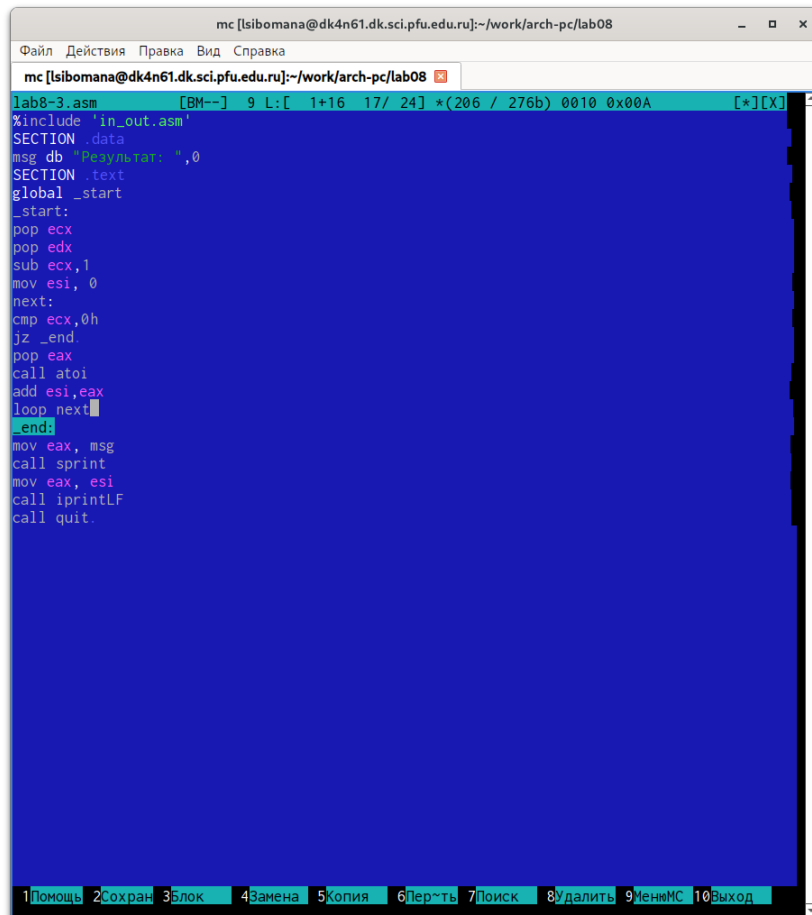
Программой было обработано 3 аргумента.

Создаем новый файл lab8-3.asm (рис. fig. ??).

```
lsibomana@dk4n61 ~/work/arch-pc/lab08 $ touch lab8-3.asm
lsibomana@dk4n61 ~/work/arch-pc/lab08 $
```

Рис. 3.7: Создание lab8-3.asm

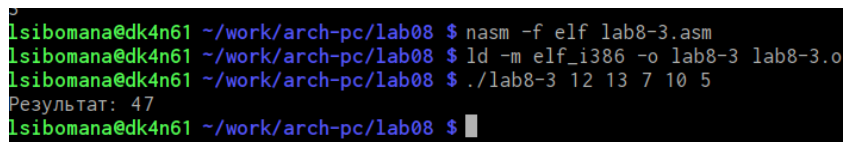
Открываем файл и заполняем его в соответствии с листингом 8.3 (рис. fig. ??)



```
lab8-3.asm [BM--] 9 L: [ 1+16 17/ 24] *(206 / 276b) 0010 0x00A [*][X]
%include "in_out.asm"
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
pop ecx
pop edx
sub ecx,1
mov esi, 0
next:
cmp ecx,0h
jz _end
pop eax
call atoi
add esi,eax
loop next
_end:
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit
```

Рис. 3.8: Текст программы lab8-3.asm

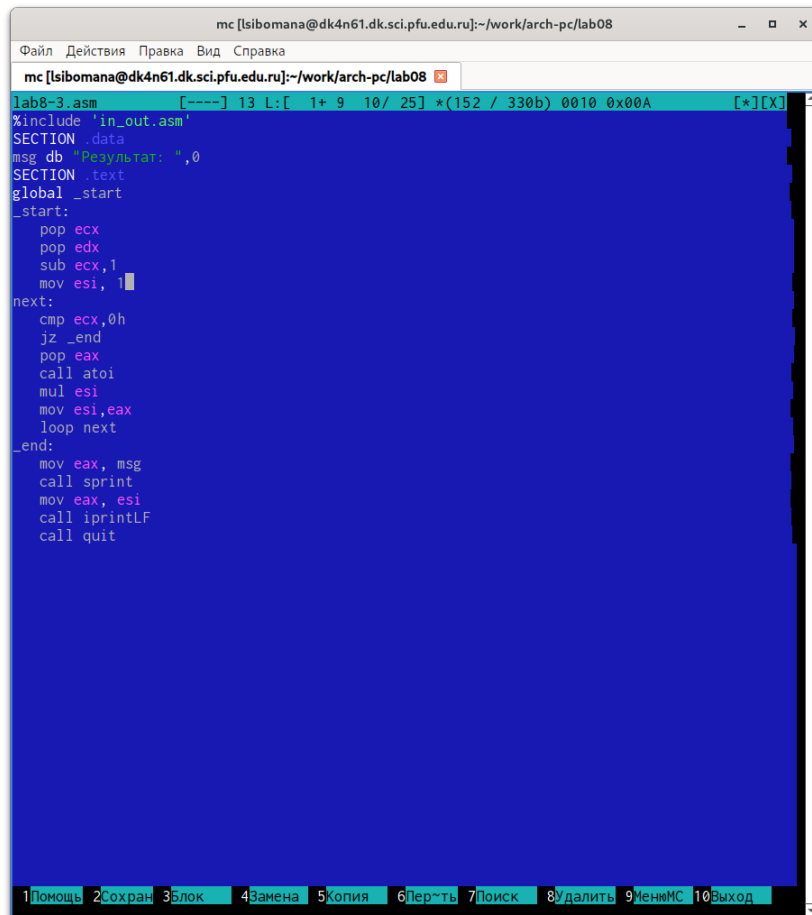
Создаём исполняемый файл и запускаем его, указав аргументы (рис. fig. ??).



```
lsibomana@dk4n61 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
lsibomana@dk4n61 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
lsibomana@dk4n61 ~/work/arch-pc/lab08 $ ./lab8-3 12 13 7 10 5
Результат: 47
lsibomana@dk4n61 ~/work/arch-pc/lab08 $
```

Рис. 3.9: Создание и запуск lab8-3.asm

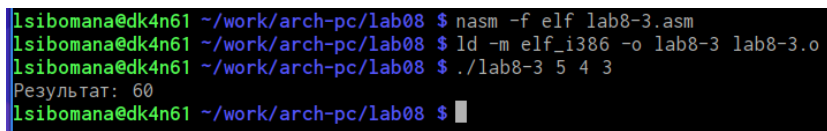
Снова открываем файл для редактирования и изменяем его, чтобы вычислялось произведение вводимых значений (рис. fig. ??)



```
lab8-3.asm [----] 13 L: [ 1+ 9 10/ 25] *(152 / 330b) 0010 0x00A [*][X]
#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
    jz _end
    pop eax
    call atoi
    mul esi
    mov esi,eax
    loop next
_end:
    mov eax,msg
    call sprint
    mov eax,esi
    call iprintLF
    call quit
```

Рис. 3.10: Текст программы lab8-3.asm

Создаём исполняемый файл и запускаем его, указав аргументы (рис. fig. ??).



```
lsibomana@dk4n61 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
lsibomana@dk4n61 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
lsibomana@dk4n61 ~/work/arch-pc/lab08 $ ./lab8-3 5 4 3
Результат: 60
lsibomana@dk4n61 ~/work/arch-pc/lab08 $
```

Рис. 3.11: Создание и запуск lab8-3.asm

ЗАДАНИЕ ДЛЯ САМОСТОЯТЕЛЬНОЙ РАБОТЫ

ВАРИАНТ-5

Напишите программу, которая находит сумму значений функции $\varphi(\varphi)$ для $\varphi = \varphi_1, \varphi_2, \dots, \varphi_n$, т.

Создаем новый файл (рис.fig. ??).

```
lsibomana@dk4n61 ~/work/arch-pc/lab08 $ touch lab8-4.asm
lsibomana@dk4n61 ~/work/arch-pc/lab08 $ mc
```

Рис. 3.12: Создание lab8-4.asm

Открываем его и пишем программу, которая выведет сумму значений, получившихся после решения выражения $4x + 3$ (рис. fig. ??).

```
lab8-4.asm [----] 35 L: [ 1+ 2 3/ 30] *(78 / 401b) 0034 0x022 [*][X]
#include 'in_out.asm'
SECTION .data
msg_func db "Функция: f(x) = 4x + 3", 0
msg_result db "Результат: ", 0
SECTION .text
GLOBAL _start
_start:
mov eax, msg_func
call sprintf
pop ecx
pop edx
sub ecx, 1
mov esi, 0
next:
cmp ecx, 0h
jz _end
pop eax
call atoi
mov ebx, 4
mul ebx
add eax, 3
add esi, eax
loop next
_end:
mov eax, msg_result
call sprintf
mov eax, esi
call iprintf
call quit
```

1 Помощь 2 Сохран 3 Блок 4 Замена 5 Копия 6 Переть 7 Поиск 8 Удалить 9 МенюМС 10 Выход

{#fig:0017w

Транслируем файл и смотрим на работу программы (рис. fig. ??).

```
lsibomana@dk4n61 ~/work/arch-pc/lab08 $ nasm -f elf lab8-4.asm
lsibomana@dk4n61 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-4 lab8-4.o
lsibomana@dk4n61 ~/work/arch-pc/lab08 $ ./lab8-4 5 6 9
Функция:  $f(x) = 4x + 3$ 
Результат: 89
lsibomana@dk4n61 ~/work/arch-pc/lab08 $
```

Рис. 3.13: Создание и запуск lab8-4.asm

Транслируем файл и смотрим на работу программы (рис. fig. ??).

```
lsibomana@dk8n60 ~/work/arch-pc $ cd lab08
lsibomana@dk8n60 ~/work/arch-pc/lab08 $ nasm -f elf lab8-4.asm
lsibomana@dk8n60 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-4 lab8-4.o
lsibomana@dk8n60 ~/work/arch-pc/lab08 $ ./lab8-4 2 4 6
Функция:  $f(x) = 4x + 3$ 
Результат: 57
lsibomana@dk8n60 ~/work/arch-pc/lab08 $
```

Рис. 3.14: Создание и запуск lab8-4.asm

4 Выводы

Мы научились решать программы с использованием циклов и обработкой аргументов командной строки.