

# **Отчёт по лабораторной работе №8**

**Отчёт по лабораторной работе №8**

Буриева Шахзода Акмаловна

# Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	8
4	Задание для самостоятельной работы	14
5	Вывод	16
	Список литературы	17

## Список иллюстраций

3.1	Создание каталога lab8,переход в него и создание файла lab8-1.asm	8
3.2	Ввод текста программы в файл lab8-1.asm . . . . .	8
3.3	Создание исполняемого файла и его запуск . . . . .	9
3.4	Изменение текста программы lab8-1.asm . . . . .	9
3.5	Создание исполняемого файла и его запуск . . . . .	9
3.6	Изменение текста программы lab8-1.asm . . . . .	10
3.7	Создание исполняемого файла и его запуск . . . . .	10
3.8	Создание файла lab8-2 . . . . .	10
3.9	Ввод текста программы в файл lab8-2.asm . . . . .	11
3.10	Создание исполняемого файла и его запуск . . . . .	11
3.11	Создание файла lab8-3 . . . . .	11
3.12	Ввод текста программы в файл lab8-3.asm . . . . .	12
3.13	Создание исполняемого файла и его запуск . . . . .	12
3.14	Изменение текста программы lab8-3.asm . . . . .	12
3.15	Создание исполняемого файла и его запуск . . . . .	13
4.1	Создание файла lab8-4 . . . . .	14
4.2	Написание программы для нахождения суммы значений функции	14
4.3	Создание исполняемого файла и его запуск . . . . .	15

## **Список таблиц**

# 1 Цель работы

Приобрести навыки 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. Команда push размещает значение в стеке, т.е. помещает значение в ячейку памяти, на которую указывает регистр esp, после этого значение регистра esp увеличивается на 4. Существует ещё две команды для добавления значений в стек. Это команда pusha, которая помещает в стек содержимое всех регистров общего назначения в следующем порядке: ax, cx, dx, bx, sp, bp, si, di. А также команда pushf, которая служит для перемещения в стек содержимого регистра флагов. Обе эти команды не имеют операндов.

Команда pop извлекает значение из стека, т.е. извлекает значение из ячейки памяти, на которую указывает регистр esp, после этого уменьшает значение регистра esp на 4. У этой команды также один операнд, который может быть регистром или переменной в памяти. Нужно помнить, что извлечённый из стека элемент не стирается из памяти и остаётся как “мусор”, который будет перезаписан при записи нового значения в стек. Для организации циклов существуют специальные инструкции. Для всех инструкций максимальное количество проходов задаётся в регистре ecx. Наиболее простой является инструкция loop. Она

позволяет организовать безусловный цикл.

Иструкция `loop` выполняется в два этапа. Сначала из регистра `ecx` вычитается единица и его значение сравнивается с нулём. Если регистр не равен нулю, то выполняется переход к указанной метке. Иначе переход не выполняется и управление передаётся команде, которая следует сразу после команды `loop`

### 3 Выполнение лабораторной работы

Создала каталог для программ лабораторной работы №8, перешла в него и создала файл lab8-1.asm.



```
saburieva@dk5n56:~/work/arch-pc/lab08
saburieva@dk5n56 ~/work/arch-pc $ mkdir ~/work/arch-pc/lab08
saburieva@dk5n56 ~/work/arch-pc $ cd ~/work/arch-pc/lab08
saburieva@dk5n56 ~/work/arch-pc/lab08 $ touch lab8-1.asm
saburieva@dk5n56 ~/work/arch-pc/lab08 $
```

Рис. 3.1: Создание каталога lab8, переход в него и создание файла lab8-1.asm

Рассмотрела пример программы вывода значений регистра ecx. Ввела в файл lab8-1.asm текст программы из листинга.



```
GNU nano 6.4 /afs/dk.sci.pfu.edu.ru/home/s/saburieva/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. 3.2: Ввод текста программы в файл lab8-1.asm

Создала исполняемый файл и запустила его. Данный пример показывает, что



использование регистра `ecx` в теле цикла `loop` может привести к некорректной работе программы. На запрос “Введите N” ввела число 6. Программа вывела числа от 1 до 6 в порядке убывания. В данном случае число проходов цикла не соответствует значению N введенному с клавиатуры.

```

sabrieva@dk556:~/work/arch-pc/lab08
sabrieva@dk556:~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
lab8-1.asm:1: error: unable to open include file 'in_out.asm': No such file or directory
sabrieva@dk556:~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
sabrieva@dk556:~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
sabrieva@dk556:~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 6
6
5
4
3
2
1
sabrieva@dk556:~/work/arch-pc/lab08 $

```

Рис. 3.3: Создание исполняемого файла и его запуск

Изменила текст программы добавив изменение значение регистра `ecx` в цикле.

```

GNU nano 2.9.2 /afs/.dk.sci.pfu.edu.ru/home/s/a/sabrieva/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:
sub ecx, 1 ; 'ecx=ecx-1'
mov [N], ecx
mov eax, [N]
call iprintf
loop label
call quit

```

Рис. 3.4: Изменение текста программы lab8-1.asm

Создала исполняемый файл и запустила его. На запрос “Введите N” ввела число 6. Программа вывела нечетные числа в порядке убывания.

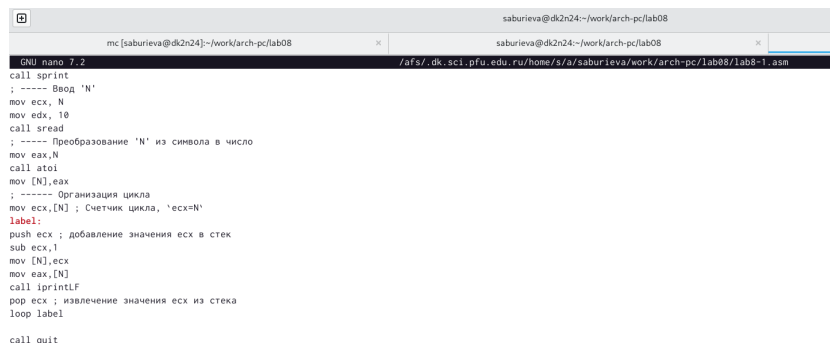
```

sabrieva@dk2n24:~/work/arch-pc/lab08
sabrieva@dk2n24:~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
sabrieva@dk2n24:~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
sabrieva@dk2n24:~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 6
5
3
1
sabrieva@dk2n24:~/work/arch-pc/lab08 $

```

Рис. 3.5: Создание исполняемого файла и его запуск

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



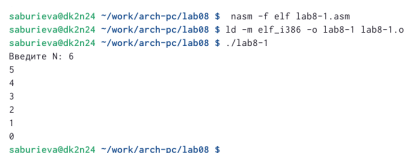
```

GNU nano 7.2
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 iprintf
pop ecx ; извлечение значения ecx из стека
loop Label
call quit

```

Рис. 3.6: Изменение текста программы lab8-1.asm

Создала исполняемый файл и запустила его. На запрос “Введите N” ввела число 6. Программа вывела числа от 0 до 6 в порядке убывания. Да, в данном случае число проходов цикла соответствует значению N введенному с клавиатуры.



```

saburieva@dk2n24 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
saburieva@dk2n24 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
saburieva@dk2n24 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 6
6
5
4
3
2
1
0
saburieva@dk2n24 ~/work/arch-pc/lab08 $

```

Рис. 3.7: Создание исполняемого файла и его запуск

Создала файл lab8-2.asm в каталоге ~/work/arch-pc/lab08.



```

saburieva@dk2n24 ~/work/arch-pc/lab08 $ touch lab8-2.asm
saburieva@dk2n24 ~/work/arch-pc/lab08 $

```

Рис. 3.8: Создание файла lab8-2

Ввела в него текст программы из листинга.

```

GNU nano 7.2 /afs/.dk.scl.pfu.edu.ru/home/s/a/saburieva/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 sprintf ; вызываем функцию печати
loop next ; переход к обработке следующего
; аргумента (переход на метку 'next')
_end:
call quit

```

Рис. 3.9: Ввод текста программы в файл lab8-2.asm

Создала исполняемый файл и запустила его, указав аргументы.

```

saburieva@dk2n24 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
saburieva@dk2n24 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
saburieva@dk2n24 ~/work/arch-pc/lab08 $ ./lab8-2 9 7 '6'
9
7
6

```

Рис. 3.10: Создание исполняемого файла и его запуск

Создала файл lab8-3.asm в каталоге ~/work/arch-pc/lab08.

```

saburieva@dk2n24 ~/work/arch-pc/lab08 $ touch lab8-3.asm
saburieva@dk2n24 ~/work/arch-pc/lab08 $

```

Рис. 3.11: Создание файла lab8-3

Ввела в него текст программы из листинга для вычисления суммы аргументов командной строки.

```

include "in_out.asm"
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
; esp ; Извлекаем из стека в 'esp' количество
; аргументов (первое значение в стеке)
; pop edx ; Извлекаем из стека в 'edx' имя программы
; ; (второе значение в стеке)
sub esp,1 ; Уменьшаем 'esp' на 1 (количество
; аргументов без названия программы)
mov esi,0 ; Используем 'esi' для хранения
; промежуточных сумм
next:
cmp esp,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 'eax+esi+eax'
loop next ; переход к обработке следующего аргумента
_end:
mov eax,msg ; вывод сообщения "Результат: "
call sprintf
mov eax,esi ; записываем сумму в регистр 'eax'
call printf ; печать результата
call quit ; завершение программы

```

Рис. 3.12: Ввод текста программы в файл lab8-3.asm

Создала исполняемый файл и запустила его, указав аргументы.

```

saborieva@dk2n24 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
saborieva@dk2n24 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
saborieva@dk2n24 ~/work/arch-pc/lab08 $ ./lab8-3 6 9 4 5 7
Результат: 31
saborieva@dk2n24 ~/work/arch-pc/lab08 $

```

Рис. 3.13: Создание исполняемого файла и его запуск

Изменила текст программы для вычисления произведения аргументов командной строки.

```

GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/s/a/saborieva/work/arch-pc/lab08/lab8-3.asm
include "in_out.asm"
SECTION .data
msg db "Результат: ",0
SECTION .text
global _start
_start:
; esp ; Извлекаем из стека в 'esp' количество
; аргументов (первое значение в стеке)
; pop edx ; Извлекаем из стека в 'edx' имя программы
; ; (второе значение в стеке)
sub esp,1 ; Уменьшаем 'esp' на 1 (количество
; аргументов без названия программы)
mov esi,1 ; Используем 'esi' для хранения
; промежуточных сумм
next:
cmp esp,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 промежуточной сумме
; след. аргумент 'esi*esi+eax'
mov esi,eax ;
loop next ; переход к обработке следующего аргумента
_end:
mov eax,msg ; вывод сообщения "Результат: "
call sprintf
mov eax,esi ; записываем сумму в регистр 'eax'
call printf ; печать результата
call quit ; завершение программы

```

Рис. 3.14: Изменение текста программы lab8-3.asm

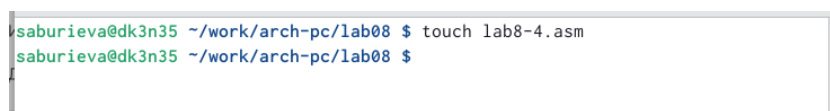
Создала исполняемый файл и запустила его, указав аргументы.

```
.. ..
saburiev@0d5n56 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
saburiev@0d5n56 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
saburiev@0d5n56 ~/work/arch-pc/lab08 $ ./lab8-3 6 7 2
Результат: 84
saburiev@0d5n56 ~/work/arch-pc/lab08 $
```

Рис. 3.15: Создание исполняемого файла и его запуск

## 4 Задание для самостоятельной работы

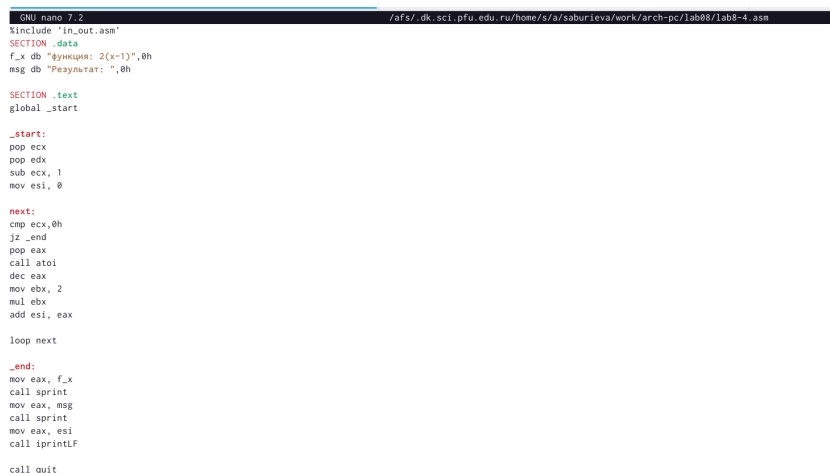
Спрева создала файл lab8-4.asm для написания программы задания самостоятельного выполнения.



```
saburieva@dk3n35 ~/work/arch-pc/lab08 $ touch lab8-4.asm
saburieva@dk3n35 ~/work/arch-pc/lab08 $
```

Рис. 4.1: Создание файла lab8-4

Написала программу, которая находит сумму значений функции  $f(x)$ . Вид функции  $f(x)$  выбрала из таблицы вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7 (вариант 4).



```
GNU nano 2.2 /afs/dk.sci.pfu.edu.ru/home/s/a/saburieva/work/arch-pc/lab08/lab8-4.asm
$include "in_out.asm"
SECTION .data
f_x db "функция: 2(x-1)",0h
msg db "Результат: ",0h
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
dec eax
mov ebx, 2
mul ebx
add esi, eax
loop next
_end:
mov eax, f_x
call sprint
mov eax, msg
call sprint
mov eax, esi
call iprintf
call quit
```

Рис. 4.2: Написание программы для нахождения суммы значений функции

Создала исполняемый файл и запустила его, указав аргументы.

```
saburiev@dk3n35 ~/work/arch-pc/lab08 $ nasm -f elf lab8-4.asm
saburiev@dk3n35 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-4 lab8-4.o
saburiev@dk3n35 ~/work/arch-pc/lab08 $ ./lab8-4 1 2 3 4
функция: 2(x-1)Результат: 12
saburiev@dk3n35 ~/work/arch-pc/lab08 $ █
```

Рис. 4.3: Создание исполняемого файла и его запуск

## 5 Вывод

Приобрела навыки написания программ с использованием циклов и обработкой аргументов командной строки.



## **Список литературы**