

# **Отчёт по лабораторной работе №8**

**Дисциплина: архитектура компьютера**

Кузнецова Елизавета Андреевна

# Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Программа для вычисления произведения аргументов командной строки	15
6	Выполнение заданий для самостоятельной работы	17
7	Программа для вычисления значения выражения $3(10+x)$	19
8	Выводы	21

# Список иллюстраций

4.1	Создание директории . . . . .	8
4.2	Создание файла . . . . .	8
4.3	Копирование файла . . . . .	8
4.4	Редактирование файла . . . . .	9
4.5	Запуск исполняемого файла . . . . .	9
4.6	Редактирование файла . . . . .	10
4.7	Создание и запуск нового исполняемого файла . . . . .	10
4.8	Редактирование файла . . . . .	11
4.9	Создание и запуск нового исполняемого файла . . . . .	11
4.10	Создание файла . . . . .	11
4.11	Редактирование файла . . . . .	12
4.12	Создание и запуск нового исполняемого файла . . . . .	12
4.13	Создание файла . . . . .	12
4.14	Редактирование файла . . . . .	13
4.15	Создание и запуск нового исполняемого файла . . . . .	13
4.16	Создание файла . . . . .	13
4.17	Редактирование файла . . . . .	14
4.18	Создание и запуск нового исполняемого файла . . . . .	14
6.1	Создание файла . . . . .	17
6.2	Написание и редактирование программы . . . . .	17
6.3	Запуск исполняемого файла . . . . .	18

## Список таблиц

# 1 Цель работы

Получение навыков по организации циклов и работе со стеком на языке NASM.

## 2 Задание

1. Ознакомиться с организацией стека.
2. Написать программу, которая находит сумму значений функции  $f(x)$  для  $x = x_1, x_2, \dots, x_n$ , т.е. программа должна выводить значение  $f(x_1) + f(x_2) + \dots + f(x_n)$ .
3. Загрузить файлы на Github.

### 3 Теоретическое введение

Стек – это специально выделенная область оперативной памяти, использующая механизм безадресной записи и выборки элементов данных. Этот механизм предполагает, что элемент, записанный последним, будет всегда прочитан первым. 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. Важно помнить, что стек растет в направлении к началу памяти и уменьшается в направлении к ее концу. Стек предназначен для временного хранения переменных, передачи параметров вызываемым подпрограммам и сохранения адреса возврата при вызове процедур и прерываний.

## 4 Выполнение лабораторной работы

С помощью утилиты `mkdir` создала директорию, в которой буду создавать файлы с программами для лабораторной работы. Перешла в созданный каталог с помощью утилиты `cd` (рис. [4.1]).

```
eakuznecova@dk8n52 ~ $ mkdir ~/work/arch-pc/lab08
eakuznecova@dk8n52 ~ $ cd ~/work/arch-pc/lab08
```

Рис. 4.1: Создание директории

С помощью утилиты `touch` создала файл `lab8-1.asm` (рис. [4.2]).

```
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ touch lab8-1.asm
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ ls
lab8-1.asm
```

Рис. 4.2: Создание файла

Скопировала в текущий каталог файл `in_out.asm` с помощью утилиты `cp`, так как он будет использоваться в других программах (рис. [4.3]).

```
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ cp ~/Зарпзкы/in_out.asm in_out.asm
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ ls
in_out.asm  lab8-1.asm
```

Рис. 4.3: Копирование файла

Открыла созданный файл `lab8-1.asm`, вставила в него программу вывода значения регистра `eax` (рис. [4.4]).



```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab08/lab8-1.asm
; Программа вывода значений регистра 'ecx'
;-----
%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. 4.4: Редактирование файла

Создала исполняемый файл программы и запустила его. Проверила его работу несколько раз(рис. [4.5]).

```
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 2
2
1
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 1
1
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
```

Рис. 4.5: Запуск исполняемого файла

Изменила текст программы файла lab8-1.asm, добавив изменение значения регистра ecx в цикле label (рис. [4.6]).

```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab08/lab8-1.asm
; Программа вывода значений регистра "ecx"
;-----
%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
loop label ; 'ecx=ecx-1' и если 'ecx' не '0'
call quit ; переход на 'label'
```

Рис. 4.6: Редактирование файла

Создала новый исполняемый файл программы и запустила его. В этой программе уменьшился изначальный индекс на 1. Получился результат, который отличается от ожидаемого. Получили  $N/2$  значений (рис. [4.7]).

```
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 10
9
7
5
3
1
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $
```

Рис. 4.7: Создание и запуск нового исполняемого файла

Изменила текст программы файла lab8-1.asm, в которую были добавлены изменения значения регистра ecx в цикле label, использовала стек (воспользовалась командами push и pop) (рис. [4.8]).

```
GNU nano 6.4 /afs/dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab08/lab8-1.asm
; Программа вывода значений регистра 'ecx'
;-----
#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 ; 'ecx=ecx-1'
mov [N],ecx
mov eax,[N]
call iprintf
pop ecx ; извлечение значения ecx из стека
loop label ; 'ecx=ecx-1' и если 'ecx' не '0'
call quit ; переход на 'label'
```

Рис. 4.8: Редактирование файла

Создала новый исполняемый файл программы и запустила его. В этой программе вывелся нужный нам результат (N значений) (рис. [4.9]).

```
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
```

Рис. 4.9: Создание и запуск нового исполняемого файла

С помощью утилиты touch создала файл lab8-2.asm (рис. [4.10]).

```
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ touch lab8-2.asm
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ ls
in_out.asm lab8-1.asm lab8-1.o lab8-2.asm
```

Рис. 4.10: Создание файла

Открыла созданный файл lab8-2.asm, вставила в него программу по обработке аргументов командной строки (рис. [4.11]).

```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab08/lab8-2.asm
; Обработка аргументов командной строки
;-----
%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. 4.11: Редактирование файла

Создала новый исполняемый файл программы и запустила его, указав аргументы. Были обработаны все введенные аргументы (рис. [4.12]).

```
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ ./lab8-2 1 2 3 4 5
1
2
3
4
5
```

Рис. 4.12: Создание и запуск нового исполняемого файла

С помощью утилиты touch создала файл lab8-3.asm (рис. [4.10]).

```
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ touch lab8-3.asm
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ ls
in_out.asm lab8-1 lab8-1.asm lab8-1.o lab8-2 lab8-2.asm lab8-2.o lab8-3.asm
```

Рис. 4.13: Создание файла

Открыла созданный файл lab8-3.asm, вставила в него программу вычисления суммы аргументов командной строки (рис. [4.14]).

```

GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/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+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. 4.14: Редактирование файла

Создала новый исполняемый файл программы и запустила его, указав аргументы. Была выведена правильная сумма аргументов (рис. [4.15]).

```

eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ ./lab8-3 1 2 3 4 5
Результат: 15

```

Рис. 4.15: Создание и запуск нового исполняемого файла

С помощью утилиты touch создала файл lab8-4.asm (рис. [4.16]).

```

eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ touch lab8-4.asm
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ ls
in_out.asm lab8-1 lab8-1.asm lab8-1.o lab8-2 lab8-2.asm lab8-2.o lab8-3 lab8-3.asm lab8-3.o lab8-4.asm

```

Рис. 4.16: Создание файла

Открыла созданный файл lab8-4.asm, вставила в него программу вычисления произведения аргументов командной строки (рис. [4.17]).

```

GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab08/lab8-4.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 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
    mul esi
    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 iprintLF ; печать результата
    call quit ; завершение программы

```

Рис. 4.17: Редактирование файла

Создала новый исполняемый файл программы и запустила его, указав аргументы. Было выведено правильное произведение аргументов (рис. [4.18]).

```

eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ nasm -f elf lab8-4.asm
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-4 lab8-4.o
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ ./lab8-4 1 2 3 4 5
Результат: 120

```

Рис. 4.18: Создание и запуск нового исполняемого файла

## 5 Программа для вычисления произведения аргументов командной строки

```
%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 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
mul esi
mov 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

```



## 6 Выполнение заданий для самостоятельной работы

Создала файл lab8-5.asm с помощью утилиты touch (рис. [6.1]).

```
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ touch lab8-5.asm
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ ls
in_out.asm lab8-1 lab8-1.asm lab8-1.o lab8-2 lab8-2.asm lab8-2.o lab8-3 lab8-3.asm lab8-3.o lab8-4 lab8-4.asm lab8-4.o lab8-5.asm
```

Рис. 6.1: Создание файла

Открыла созданный файл и ввела в него текст программы, которая находит сумму значений функции  $f(x)=3(10+x)$ , которое было под 20 вариантом (рис. [6.2]).

```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab08/lab8-5.asm
#include "in_out.asm"

SECTION .data
f_x db "функция: 3(10+x)",0h
msg db 10, "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
add eax, 10
mov ebx, 3
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. 6.2: Написание и редактирование программы

Создала новый исполняемый файл и запустила его. Проверила свою программу

на трех наборах  $x$ . Все значения выводятся верно. (рис. [6.3]).

```
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ nasm -f elf lab8-5.asm
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-5 lab8-5.o
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ ./lab8-5 1 6 7 8
функция: 3(10+x)
результат: 186
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ ./lab8-5 1 2 3 4 5
функция: 3(10+x)
результат: 195
eakuznecova@dk8n52 ~/work/arch-pc/lab08 $ ./lab8-5 6 9 10 12 13
функция: 3(10+x)
результат: 300
```

Рис. 6.3: Запуск исполняемого файла

## 7 Программа для вычисления значения выражения $3(10+x)$

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
f_x db "функция: 3(10+x)",0h
```

```
msg db 10, '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
```

```
add eax, 10
mov ebx, 3
mul ebx
add esi, eax

loop next

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

call quit
```

## 8 Выводы

В ходе этой лабораторной работы были получены навыки по организации циклов и работе со стеком на языке NASM.