

Отчёт по лабораторной работе №8

Дисциплина: Архитектура компьютера

Абакумов Тимофей Александрович

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Реализация циклов в NASM	7
3.2	Обработка аргументов командной строки	13
3.3	Заданий для самостоятельной работы	19
4	Выводы	23

Список иллюстраций

3.1	Перемещение между директориями	7
3.2	Код программы	8
3.3	Работа программы	9
3.4	Изменение программы	10
3.5	Работа файла	10
3.6	Изменение программы	12
3.7	Работа файла	13
3.8	Создание файла	13
3.9	Код программы	14
3.10	Работа файла	15
3.11	Создание файла	15
3.12	Код программы	16
3.13	Работа файла	17
3.14	Изменение программы	18
3.15	Работа файла	19
3.16	Создание файла	20
3.17	Код программы	20
3.18	Код программы	22

Список таблиц

1 Цель работы

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

2 Задание

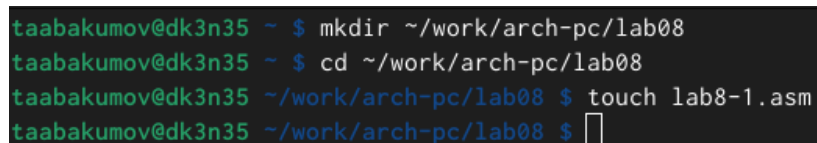
Порядок выполнения лабораторной работы

1. Реализация циклов в NASM.
2. Обработка аргументов командной строки.
3. Выполнение заданий для самостоятельной работы

3 Выполнение лабораторной работы

3.1 Реализация циклов в NASM

1. Для начала создадим каталог для программ лабораторной работы № 8, перейдём в него и создадим файл lab8-1.asm (рис. 3.1).



```
taabakumov@dk3n35 ~ $ mkdir ~/work/arch-pc/lab08
taabakumov@dk3n35 ~ $ cd ~/work/arch-pc/lab08
taabakumov@dk3n35 ~/work/arch-pc/lab08 $ touch lab8-1.asm
taabakumov@dk3n35 ~/work/arch-pc/lab08 $
```

Рис. 3.1: Перемещение между директориями

2. Введём в файл lab8-1.asm текст программы из листинга 8.1 (рис. 3.2).

```

lab8-1.asm      [----]  9 L:[  1+ 0   1/ 32] *(9   / 854b) 0045 0x02D
;-----[ ]-----
; 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

```

Рис. 3.2: Код программы

Код программы из пункта 2:

```

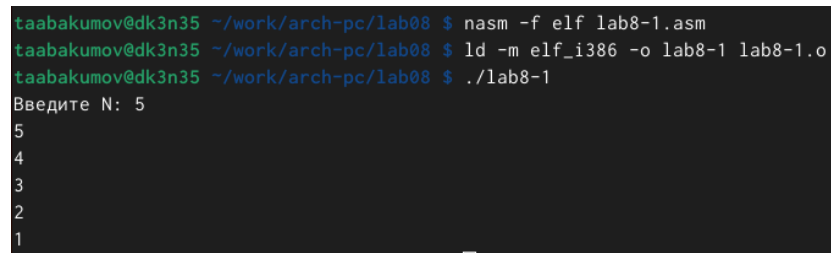
#include "in_out.asm"
SECTION .data
msg db 'введите N:', 0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
mov eax,msg
call sprint
mov ecx,N
mov edx,10
call sread
mov eax,N
call atoi

```



```
mov [N],eax
mov ecx,[N]
label:
mov [N],ecx
mov eax, [N]
call iprintLF
loop label
call quit
```

3. Создадим исполняемый файл и проверим его работу (рис. 3.3).



```
taabakumov@dk3n35 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
taabakumov@dk3n35 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
taabakumov@dk3n35 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 5
5
4
3
2
1
```

Рис. 3.3: Работа программы

4. Данный пример показывает, что использование регистра ecx в теле цикла loop может привести к некорректной работе программы. Изменим текст программы добавив изменение значение регистра ecx в цикл (рис. 3.4).

```

mov ecx,[N]

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

call quit

```

Рис. 3.4: Изменение программы

5. Создадим исполняемый файл и проверим его работу (рис. 3.5).

```

taabakumov@dk3n35 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
taabakumov@dk3n35 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
taabakumov@dk3n35 ~/work/arch-pc/lab08 $ ./lab8-1
введите N: 10
9
7
5
3
1
taabakumov@dk3n35 ~/work/arch-pc/lab08 $ 

```

Рис. 3.5: Работа файла

Код программы из пункта 5:

```

#include "in_out.asm"

SECTION .data
msg db 'введите N:', 0h

```

```

SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
mov eax,msg
call sprint
mov ecx,N
mov edx,10
call sread
mov eax,N
call atoi
mov [N],eax
mov ecx,[N]
label:
sub ecx,1
mov [N],ecx
mov eax, [N]
call iprintLF
loop label
call quit

```

6. Для использования регистра `ecx` в цикле и сохранения корректности работы программы можно использовать стек. Внесём изменения в текст программы добавив команды `push` и `pop` (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла `loop` (рис. 3.6).

```

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

call quit

```

Рис. 3.6: Изменение программы

Код программы из пункта 6:

```

#include "in_out.asm"

SECTION .data
msg db 'введите N:', 0h

SECTION .bss
N: resb 10

SECTION .text
global _start
_start:
mov eax,msg
call sprint
mov ecx,N
mov edx,10
call sread
mov eax,N
call atoi

```

```

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

```

7. Создадим исполняемый файл и проверим его работу (рис. 3.7).



```

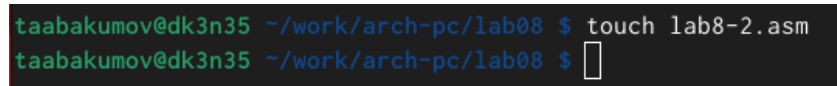
taabakumov@dk3n35 ~/work/arch-pc/lab08 $ nasm -f elf lab8-1.asm
taabakumov@dk3n35 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-1 lab8-1.o
taabakumov@dk3n35 ~/work/arch-pc/lab08 $ ./lab8-1
введите N: 10
9
8
7
6
5
4
3
2
1
0

```

Рис. 3.7: Работа файла

3.2 Обработка аргументов командной строки

8. Создадим файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и введём в него текст программы из листинга 8.2 (рис. 3.8).



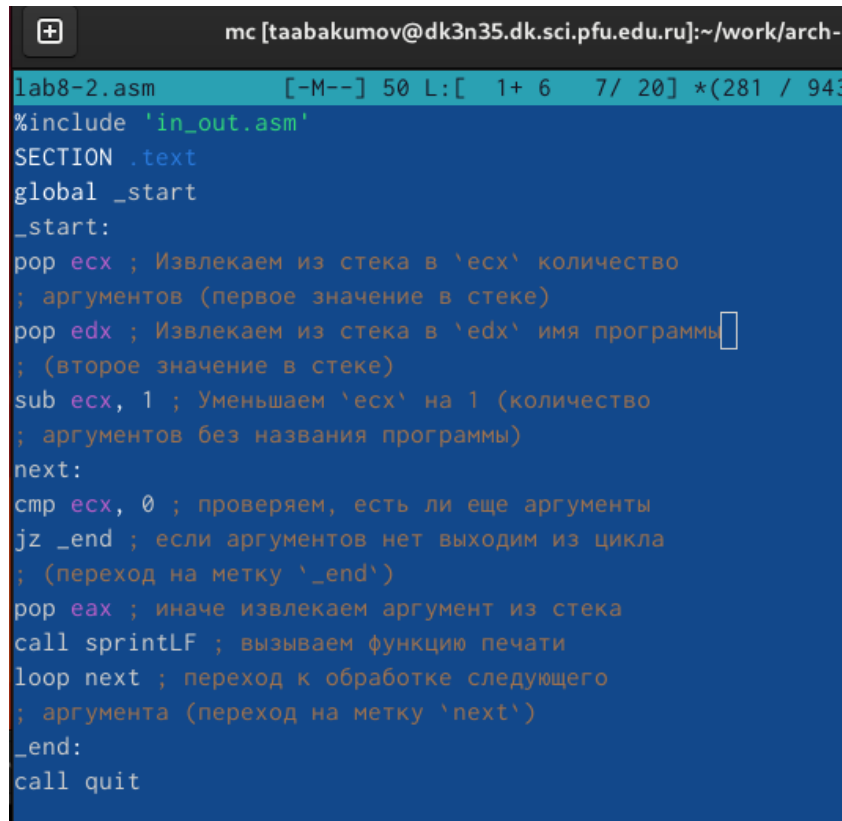
```

taabakumov@dk3n35 ~/work/arch-pc/lab08 $ touch lab8-2.asm
taabakumov@dk3n35 ~/work/arch-pc/lab08 $ █

```

Рис. 3.8: Создание файла

9. Введём в него текст программы из листинга 8.2 (рис. 3.9).



```
lab8-2.asm      [-M--] 50 L:[ 1+ 6 7/ 20] *(281 / 943
#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: Код программы

Код программы из пункта 9:

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

```
SECTION .text
```

```
global _start
```

```
_start:
```

```
pop ecx
```

```
pop edx
```

```
sub ecx, 1
```

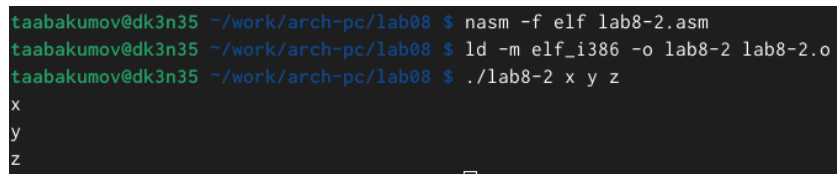
```
next:
```

```
cmp ecx, 0
```

```
jz _end
```

```
pop eax
call sprintLF
loop next
_end:
call quit
```

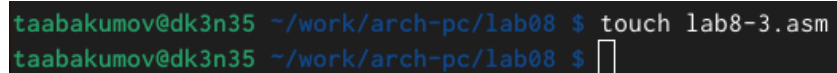
10. Создадим исполняемый файл и запустим его, указав аргументы (рис. 3.10).



```
taabakumov@dk3n35 ~/work/arch-pc/lab08 $ nasm -f elf lab8-2.asm
taabakumov@dk3n35 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-2 lab8-2.o
taabakumov@dk3n35 ~/work/arch-pc/lab08 $ ./lab8-2 x y z
x
y
z
```

Рис. 3.10: Работа файла

11. Рассмотрим еще один пример программы которая выводит сумму чисел, которые передаются в программу как аргументы. Создадим файл lab8-3.asm в каталоге ~/work/arch-pc/lab08 (рис. 3.11).



```
taabakumov@dk3n35 ~/work/arch-pc/lab08 $ touch lab8-3.asm
taabakumov@dk3n35 ~/work/arch-pc/lab08 $
```

Рис. 3.11: Создание файла

12. Введём в него текст программы из листинга 8.3 (рис. 3.12).

```

lab8-3.asm      [-M--]  7 L:[  1+ 5   6/ 29] *(103 /142
%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'
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ть 7Поиск 8

```

Рис. 3.12: Код программы

Код программы из пункта 12:

```

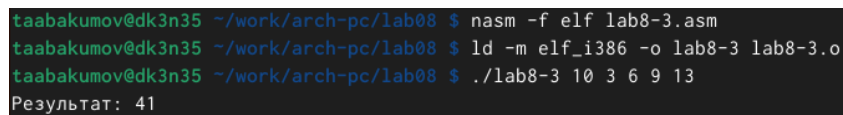
%include 'in_out.asm'
SECTION .data
msg db "Результат:",0
SECTION .text
global _start
_start:
pop ecx
pop edx
sub ecx,1
mov esi, 0
next:
cmp ecx,0h

```



```
jz _end
pop eax
call atoi
add esi,eax
loop next
_end:
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit
```

13. Создадим исполняемый файл и запустим его, указав аргументы (рис. 3.13).



```
taabakumov@dk3n35 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
taabakumov@dk3n35 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
taabakumov@dk3n35 ~/work/arch-pc/lab08 $ ./lab8-3 10 3 6 9 13
Результат: 41
```

Рис. 3.13: Работа файла

14. Изменим текст программы из листинга 8.3 для вычисления произведения аргументов командной строки (рис. 3.14).

```

lab8-3.asm      [----]  5 L:[  3+10
msg db 'результат: '
SECTION .text
GLOBAL _start

_start:
pop ecx
pop edx
sub ecx,1
mov esi,1

next:
cmp ecx,0
jz _end

pop eax
call atoi
mul esi
mov esi, eax

```

Рис. 3.14: Изменение программы

Код программы из пункта 14:

```
%include "in_out.asm"
```

```
SECTION .data
```

```
msg db 'результат:'
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
pop ecx
```

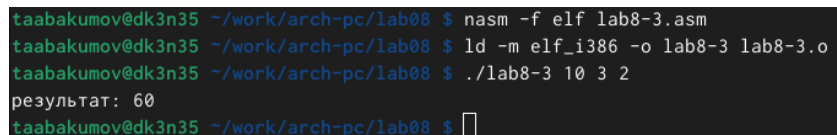
```
pop edx
```

```

sub ecx,1
mov esi,1
next:
cmp ecx,0
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

```

15. Создадим исполняемый файл и запустим его, указав аргументы (рис. 3.15).



```

taabakumov@dk3n35 ~/work/arch-pc/lab08 $ nasm -f elf lab8-3.asm
taabakumov@dk3n35 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 lab8-3.o
taabakumov@dk3n35 ~/work/arch-pc/lab08 $ ./lab8-3 10 3 2
результат: 60
taabakumov@dk3n35 ~/work/arch-pc/lab08 $ █

```

Рис. 3.15: Работа файла

3.3 Заданий для самостоятельной работы

16. Для начала создадим файл lab8-4.asm в каталоге ~/work/arch-pc/lab08 (рис. 3.16).

```
taabakumov@dk3n55 ~/work/arch-pc/lab08 $ touch lab8-4.asm
taabakumov@dk3n55 ~/work/arch-pc/lab08 $
```

Рис. 3.16: Создание файла

17. Напишем программу, которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, \dots, x_n$, т.е. программа должна выводить значение $f(x_1) + f(x_2) + \dots + f(x_n)$. Значения x_i передаются как аргументы. Вид функции $f(x)$ выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Мой вариант - 9 (рис. 3.17).

```
lab8-4.asm [----] 9 L:[ 1+ 0 1/ 40] *(9 / 360b) 0039 0x027
#include 'in_out.asm'

SECTION .data
prim DB 'f(x)=10x-4',0
otv DB 'Результат: ',0
SECTION .text
GLOBAL _start
_start:

pop ecx

pop edx

sub ecx,1

mov esi,0

mov eax,prim
call sprintf
next:
cmp ecx,0
jz _end
```

Рис. 3.17: Код программы

Код программы из пункта 17:

```
%include 'in_out.asm'
SECTION .data
prim DB 'f(x)=10x-4',0
otv DB 'Результат:',0
SECTION .text
GLOBAL _start
```

```

_start:
pop ecx
pop edx
sub ecx,1
mov esi,0
mov eax,prim
call sprintLF
next:
cmp ecx,0
jz _end
mov ebx,10
pop eax
call atoi
mul ebx
add eax,-4
add esi,eax
loop next
_end:
mov eax,otv
call sprint
mov eax,esi
call iprintLF
call quit

```

18. Создадим исполняемый файл и проверим его работу на нескольких наборах $x = x_1, x_2, \dots, x$ (рис. 3.18).

```
taabakumov@dk3n55 ~/work/arch-pc/lab08 $ nasm -f elf lab8-4.asm
taabakumov@dk3n55 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-4 lab8-4.o
taabakumov@dk3n55 ~/work/arch-pc/lab08 $ ./lab8-4 1
f(x)=10x-4
Результат: 6
taabakumov@dk3n55 ~/work/arch-pc/lab08 $ ./lab8-4 2
f(x)=10x-4
Результат: 16
```

Рис. 3.18: Код программы

4 Выводы

Я приобрел навыки написания программы с использованием цикла.