

Лабораторная работа 8

Мазуркевич Анастасия Дмитриевна

Содержание

Цель работы	5
Выполнение лабораторной работы	6
Реализация циклов в NASM	6
Обработка аргументов командной строки	10
Задание для самостоятельной работы	12
Выводы	14

Список иллюстраций

1	Создаем каталог и файл	6
2	Вводим листинг	7
3	Создаем файл и проверяем работу	7
4	Меняем текст	8
5	Создаем исполняемый файл и проверяем	8
6	Вносим изменения	9
7	Создаем и проверяем	9
8	Создаем файл	10
9	Программа	10
10	Проверяем	11
11	Создаем файл и вводим листинг	11
12	Создаем файл и проверяем	11
13	Изменяем текст	12
14	Все верно	12
15	Создаем	12
16	Программа	13
17	Проверяем работу первый раз	13
18	Проверяем работу второй раз	13

Список таблиц

Цель работы

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

Выполнение лабораторной работы

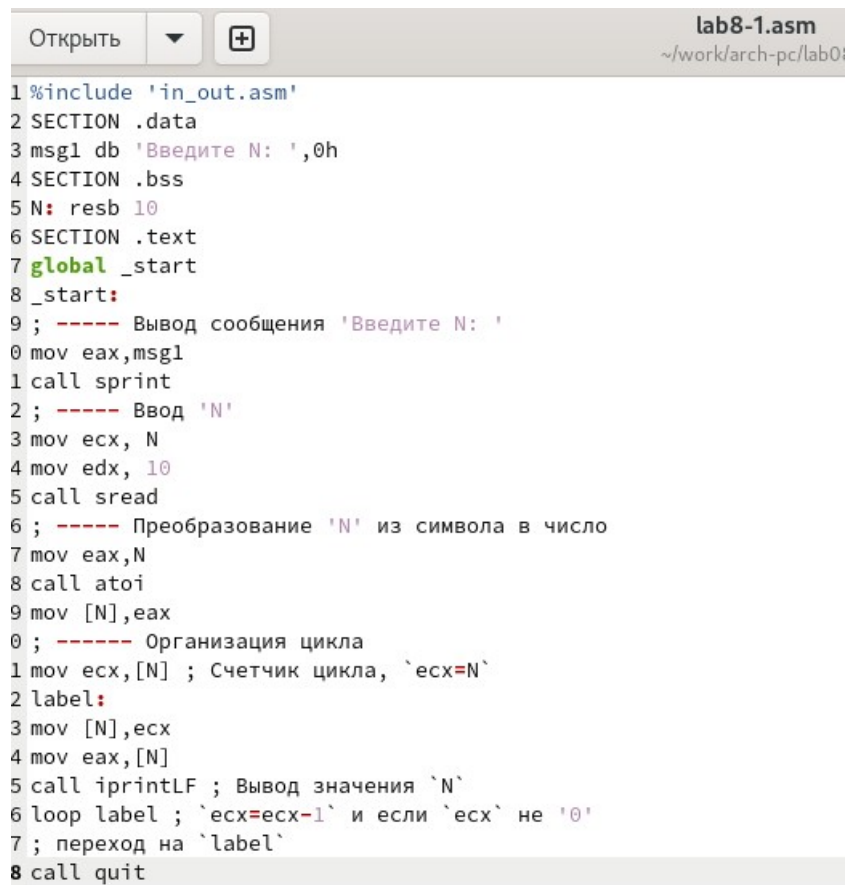
Реализация циклов в NASM

Создайте каталог для программ лабораторной работы № 8, перейдите в него и создайте файл lab8-1.asm: (рис. [-@fig:001]).

```
report$ mkdir ~/work/arch-pc/lab08
amazurkevich@vbox:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs/lab07/
report$ cd ~/work/arch-pc
amazurkevich@vbox:~/work/arch-pc$ ls
lab04 lab05 lab06 lab07 lab08
amazurkevich@vbox:~/work/arch-pc$ cd lab08
amazurkevich@vbox:~/work/arch-pc/lab08$ touch lab8-1.asm
amazurkevich@vbox:~/work/arch-pc/lab08$
```

Рис. 1: Создаем каталог и файл

Введите в файл lab8-1.asm текст программы из листинга 8.1.




```
lab8-1.asm
~/work/arch-pc/lab01

1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db 'Введите N: ',0h
4 SECTION .bss
5 N: resb 10
6 SECTION .text
7 global _start
8 _start:
9 ; ----- Вывод сообщения 'Введите N: '
10 mov eax,msg1
11 call sprint
12 ; ----- Ввод 'N'
13 mov ecx, N
14 mov edx, 10
15 call sread
16 ; ----- Преобразование 'N' из символа в число
17 mov eax,N
18 call atoi
19 mov [N],eax
20 ; ----- Организация цикла
21 mov ecx,[N] ; Счетчик цикла, `ecx=N`
22 label:
23 mov [N],ecx
24 mov eax,[N]
25 call iprintLF ; Вывод значения `N`
26 loop label ; `ecx=ecx-1` и если `ecx` не `0`
27 ; переход на `label`
28 call quit
```

Рис. 2: Вводим листинг

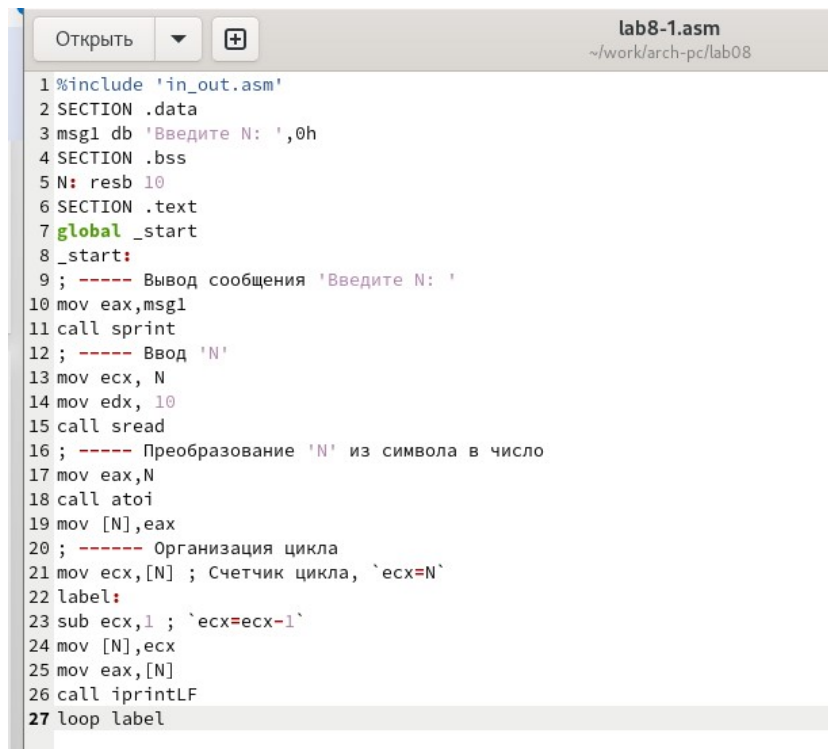
Создайте исполняемый файл и проверьте его работу



```
Введите N: 8
8
7
6
5
4
3
2
1
amazurkevich@vbox:~/work/arch-pc/lab08$
```

Рис. 3: Создаем файл и проверяем работу

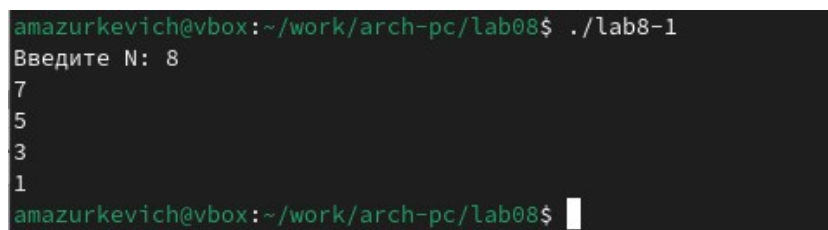
Измените текст программы добавив изменение значение регистра ecx в цикле:



```
1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db 'Введите N: ',0h
4 SECTION .bss
5 N: resb 10
6 SECTION .text
7 global _start
8 _start:
9 ; ----- Вывод сообщения 'Введите N: '
10 mov eax,msg1
11 call sprint
12 ; ----- Ввод 'N'
13 mov ecx, N
14 mov edx, 10
15 call sread
16 ; ----- Преобразование 'N' из символа в число
17 mov eax,N
18 call atoi
19 mov [N],eax
20 ; ----- Организация цикла
21 mov ecx,[N] ; Счетчик цикла, `ecx=N`
22 label:
23 sub ecx,1 ; `ecx=ecx-1`
24 mov [N],ecx
25 mov eax,[N]
26 call iprintLF
27 loop label
```

Рис. 4: Меняем текст

Создайте исполняемый файл и проверьте его работу. Какие значения принимает регистр `ecx` в цикле? Соответствует ли число проходов цикла значению `N` введенному с клавиатуры? Принимает значения 7 5 3 1 и число проходов не равно `N`



```
amazurkevich@vbox:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 8
7
5
3
1
amazurkevich@vbox:~/work/arch-pc/lab08$
```

Рис. 5: Создаем исполняемый файл и проверяем

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


```

1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db 'Введите N: ',0h
4 SECTION .bss
5 N: resb 10
6 SECTION .text
7 global _start
8 _start:
9 ; ----- Вывод сообщения 'Введите N: '
10 mov eax,msg1
11 call sprint
12 ; ----- Ввод 'N'
13 mov ecx, N
14 mov edx, 10
15 call sread
16 ; ----- Преобразование 'N' из символа в число
17 mov eax,N
18 call atoi
19 mov [N],eax
20 ; ----- Организация цикла
21 mov ecx,[N] ; Счетчик цикла, `ecx=N`
22 label:
23 push ecx
24 sub ecx,1
25 mov [N],ecx
26 mov eax,[N]
27 call iprintLF ; Вывод значения `N`
28 pop ecx
29 loop label ; `ecx=ecx-1` и если `ecx` не `0`
30 ; переход на `label`
31 call quit

```

Рис. 6: Вносим изменения

Создайте исполняемый файл и проверьте его работу. Соответствует ли в данном случае число проходов цикла значению \square введенному с клавиатуры? Да, число проходов равно числу N

```

Введите N: 8
7
6
5
4
3
2
1
0
amazurkevich@vbox:~/work/arch-pc/lab08$

```

Рис. 7: Создаем и проверяем

Обработка аргументов командной строки

Создайте файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и введите в него текст программы из листинга 8.2

```
amazurkevich@vbox:~/work/arch-pc/lab08$ touch lab8-2.asm  
amazurkevich@vbox:~/work/arch-pc/lab08$
```

Рис. 8: Создаем файл

Пишем программу из листинга

```
1 %include 'in_out.asm'  
2 SECTION .text  
3 global _start  
4 _start:  
5 pop ecx ; Извлекаем из стека в `ecx` количество  
6 ; аргументов (первое значение в стеке)  
7 pop edx ; Извлекаем из стека в `edx` имя программы  
8 ; (второе значение в стеке)  
9 sub ecx, 1 ; Уменьшаем `ecx` на 1 (количество  
10 ; аргументов без названия программы)  
11 next:  
12 cmp ecx, 0 ; проверяем, есть ли еще аргументы  
13 jz _end ; если аргументов нет выходим из цикла  
14 ; (переход на метку `_end`)  
15 pop eax ; иначе извлекаем аргумент из стека  
16 call sprintf ; вызываем функцию печати  
17 loop next ; переход к обработке следующего  
18 ; аргумента (переход на метку `next`)  
19 _end:  
20 call quit
```

Рис. 9: Программа

Создайте исполняемый файл и запустите его, указав аргументы. Программа обработала 3 аргумента

```
amazurkevich@vbox: ~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
amazurkevich@vbox: ~/work/arch-pc/lab08$ ./lab8-2
amazurkevich@vbox: ~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
amazurkevich@vbox: ~/work/arch-pc/lab08$
```

Рис. 10: Проверяем

Создайте файл lab8-3.asm в каталоге ~/work/archpc/lab08 и введите в него текст программы из листинга 8.3.

```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
6 _start:
7 pop ecx ; Извлекаем из стека в `ecx` количество
8 ; аргументов (первое значение в стеке)
9 pop edx ; Извлекаем из стека в `edx` имя программы
10 ; (второе значение в стеке)
11 sub ecx,1 ; Уменьшаем `ecx` на 1 (количество
12 ; аргументов без названия программы)
13 mov esi, 0 ; Используем `esi` для хранения
14 ; промежуточных сумм
15 next:
16 cmp ecx,0h ; проверяем, есть ли еще аргументы
17 jz _end ; если аргументов нет выходим из цикла
18 ; (переход на метку `_end`)
19 pop eax ; иначе извлекаем следующий аргумент из стека
20 call atoi ; преобразуем символ в число
21 add esi,eax ; добавляем к промежуточной сумме
22 ; след. аргумент `esi=esi+eax`
23 loop next ; переход к обработке следующего аргумента
24 _end:
25 mov eax, msg ; вывод сообщения "Результат: "
26 call sprint
27 mov eax, esi ; записываем сумму в регистр `eax`
28 call iprintLF ; печать результата
29 call quit ; завершение программы
```

Рис. 11: Создаем файл и вводим листинг

Создайте исполняемый файл и запустите его, указав аргументы.

```
amazurkevich@vbox: ~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
amazurkevich@vbox: ~/work/arch-pc/lab08$
```

Рис. 12: Создаем файл и проверяем

Измените текст программы из листинга 8.3 для вычисления произведения аргументов командной строки.

```

1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .text
5 global _start
6 _start:
7     pop ecx
8     pop edx
9     sub ecx,1
10    mov esi, 1
11 next:
12    cmp ecx,0h
13    jz _end
14    pop eax
15    call atoi
16    mul esi
17    mov esi,eax
18    loop next
19 _end:
20    mov eax,msg
21    call sprint
22    mov eax,esi
23    call iprintLF
24    call quit

```

Рис. 13: Изменяем текст

Проверяем работу, все верно

```

amazurkevich@vbox:~/work/arch-pc/lab08$ ./lab8-3 2 4 6
Результат: 48

```

Рис. 14: Все верно

Задание для самостоятельной работы

Создаем файл

```

amazurkevich@vbox:~/work/arch-pc/lab08$ touch lab8-4.asm
amazurkevich@vbox:~/work/arch-pc/lab08$ 
mov eax,msg

```

Рис. 15: Создаем

Пишем программу

```
1 %include 'in_out.asm'
2 SECTION .data
3 msg db "Результат: ",0
4 SECTION .bss
5 prm: RESB 80
6 SECTION .text
7 global _start
8 _start:
9     pop ecx
10    pop edx
11    sub ecx,1
12    mov esi,2
13 next:
14    cmp ecx,0h
15    jz _end
16    pop eax
17    call atoi
18    mul esi
19    sub eax,2
20    add eax,ebx
21    add [prm],eax
22    loop next
23 _end:
24    mov eax,msg
25    call sprint
26    mov eax,[prm]
27    call iprintLF
28    call quit
```

Рис. 16: Программа

Создаем файл и запускаем

```
amazurkevich@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
amazurkevich@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
amazurkevich@vbox:~/work/arch-pc/lab08$ ./lab8-4 12 14 18
Результат: 82
```

Рис. 17: Проверяем работу первый раз

```
amazurkevich@vbox:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
amazurkevich@vbox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
amazurkevich@vbox:~/work/arch-pc/lab08$ ./lab8-4 14 22 15
Результат: 96
amazurkevich@vbox:~/work/arch-pc/lab08$
```

Рис. 18: Проверяем работу второй раз

Выводы

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