

Отчёт по лабораторной работе №8

Дисциплина: Архитектура компьютера

Максимова Дарья Валерьевна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Обработка аргументов командной строки	11
3.2	Задание для самостоятельной работы	15
4	Выводы	18

Список иллюстраций

3.1	Текст программы	7
3.2	Работа программы	8
3.3	Изменения в программе	8
3.4	Работа программы	10
3.5	Работа программы после изменений	11
3.6	Текст программы	12
3.7	Работа программы	12
3.8	Текст программы	13
3.9	Результат	13
3.10	Текст программы	14
3.11	Результаты	14
3.12	Код моей программы	16
3.13	Работает успешно!	17

Список таблиц

1 Цель работы

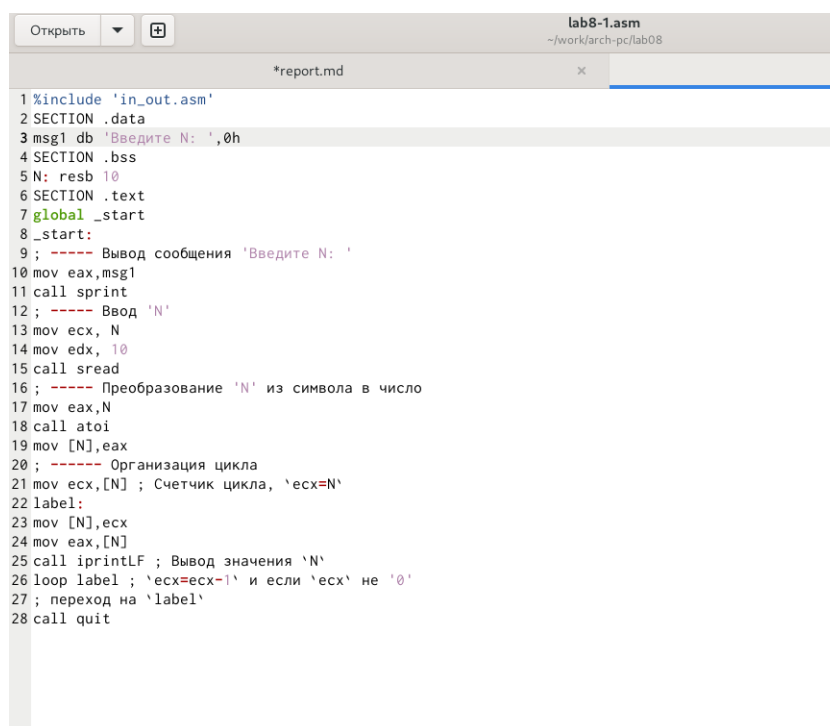
Приобретение навыков написания программ с использованием циклов и обработкой аргументов командной строки.

2 Задание

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

3 Выполнение лабораторной работы

Для начала создаю каталог для выполнения лаб. работы и в нем создаю файл lab08-1.asm. Далее ввожу в этот файл текст из листинга 8.1: (рис. 3.1).



```
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. 3.1: Текст программы

Вот как работает эта программа: (рис. 3.2).

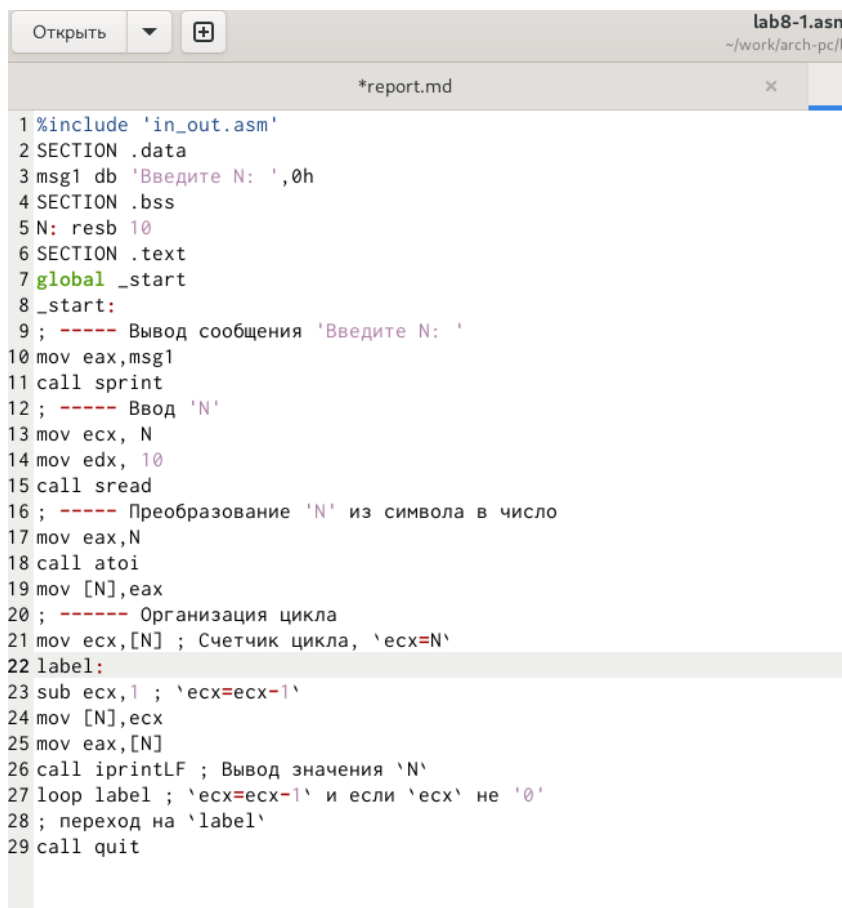
```

dvmaksimova@dk8n64 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 8
8
7
6
5
4
3
2
1
dvmaksimova@dk8n64 ~/work/arch-pc/lab08 $ █

```

Рис. 3.2: Работа программы

Добавляю изменения в текст программы согласно методическим материалам (рис. 3.3).



```

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

*report.md

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 ; Вывод значения '\N'
27 loop label ; 'ecx=ecx-1' и если 'ecx' не '0'
28 ; переход на 'label'
29 call quit

```

Рис. 3.3: Изменения в программе

Вот такой пугающий результат работы программы(рис. 3.4).

```
mc [dvmaksimova@dk8n64.dk.sci.pfu.edu.ru]:~/work/
4294754576
4294754574
4294754572
4294754570
4294754568
4294754566
4294754564
4294754562
4294754560
4294754558
4294754556
4294754554
4294754552
4294754550
4294754548
4294754546
4294754544
4294754542
4294754540
4294754538
4294754536
4294754534
4294754532
4294754530
4294754528
4294754526
4294754524
4294754522
4294754520
4294754518
4294754516
4294754514
4294754512
4294754510
4294754508
4294754506
4294754504
4294754502
42947^Z
[1]+  Остановлен      ./lab8-1
dvmaksimova@dk8n64 ~/work/arch-pc/lab08 $ █
```

Рис. 3.4: Работа программы

Как мы видим, программа выводит огромное количество чисел, и, таким образом число проходов цикла гораздо больше чем введенное N.

Я вношу изменения в программу. добавляя команды push и pop. Запускаю программу и теперь все стало нормально. Количество проходов цикла соответствует введенному мной N(рис. 3.5).

```
dvmaksimova@dk8n64 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 2
1
0
dvmaksimova@dk8n64 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 6
5
4
3
2
1
0
dvmaksimova@dk8n64 ~/work/arch-pc/lab08 $ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
dvmaksimova@dk8n64 ~/work/arch-pc/lab08 $
```

Рис. 3.5: Работа программы после изменений

3.1 Обработка аргументов командной строки

Создаю новый файл и ввожу в него текст листинга 8.2 (рис. 3.6).

```

report.n
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 sprintLF ; вызываем функцию печати
17 loop next ; переход к обработке следующего
18 ; аргумента (переход на метку `next`)
19 _end:
20 call quit

```

Рис. 3.6: Текст программы

Запускаю файл в работу, указав аргументы (рис. 3.7).

```

dvmaksimova@dk8n64 - dvmaksimova x dvmaksimova@dk8n64 - lab08 x
dvmaksimova@dk8n64 ~/work/arch-pc/lab08 $ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
dvmaksimova@dk8n64 ~/work/arch-pc/lab08 $

```

Рис. 3.7: Работа программы

Можно сказать, что программой было обработано 4 аргумента.

Для рассмотрения следующего примера я создаю новый файл и ввожу туда текст листинга 8.3(рис. 3.8).

```

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. 3.8: Текст программы

Результат работы получился соответствующим (рис. 3.9).

```

dvmaksimova@dk8n64 ~/work/arch-pc/lab08 $ ./lab8-3 12 13 7 10 5
Результат: 47
dvmaksimova@dk8n64 ~/work/arch-pc/lab08 $

```

Рис. 3.9: Результат

Теперь нам нужно изменить текст программы под аргументы(рис. 3.10).

```

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, 1 ; Используем 'esi' для хранения
14 ; промежуточных сумм
15 next:
16
17 cmp ecx,0h ; проверяем, есть ли еще аргументы
18 jz _end ; если аргументов нет выходим из цикла
19 ; (переход на метку '_end')
20
21 pop eax ; иначе извлекаем следующий аргумент из стека
22 call atoi ; преобразуем символ в число
23 mul esi ; добавляем к промежуточной сумме
24 mov esi,eax
25 ; след. аргумент 'esi=esi+eax'
26 loop next ; переход к обработке следующего аргумента
27 _end:
28 mov eax, msg ; вывод сообщения "Результат: "
29 call sprint
30 mov eax, esi ; записываем сумму в регистр 'eax'
31 call iprintLF ; печать результата
32 call quit ; завершение программы

```

Рис. 3.10: Текст программы

Как видим текст программы изменен успешно и у нас все работает (рис. 3.11).

```

dvmaksimova@dk8n64 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o lab8-3 l
ab8-3.o
dvmaksimova@dk8n64 ~/work/arch-pc/lab08 $ ./lab8-3 12 13 7 10 5
Результат: 54600
dvmaksimova@dk8n64 ~/work/arch-pc/lab08 $ ./lab8-3 4 10
Результат: 40
dvmaksimova@dk8n64 ~/work/arch-pc/lab08 $ ./lab8-3 2 3 5
Результат: 30
dvmaksimova@dk8n64 ~/work/arch-pc/lab08 $ 

```

Рис. 3.11: Результаты

3.2 Задание для самостоятельной работы

Для выполнения самостоятельного задания я создам файл `poter.asm` в том же каталоге, где выполняла предыдущие задания. Суть задания заключается в том, что я должна самостоятельно написать такую программу, которая будет выводить сумму значений функций исходя из того, какие переменные я буду вводить в программу (как аргументы). При выполнении лабораторной работы №7, я выполняла задания для 2 варианта, соответственно при выполнении этой лабораторной работы, я тоже буду выполнять 2 вариант.

Самостоятельно пишу код для того, чтоб программа успешно работала:(рис. 3.12).

```

1 %include 'in_out.asm'
2 SECTION .data
3 msg1 db "Функция: f(x)=3x-1", 0
4 msg2 db "Результат: ", 0
5 SECTION .text
6 global _start
7 _start:
8
9 pop ecx
10 pop edx
11 sub ecx, 1
12 mov esi, 0
13
14 next:
15 cmp ecx, 0h
16 jz _end
17 pop eax
18 call atoi
19
20 mov ebx, eax
21 add eax, ebx
22 add eax, ebx
23 sub eax, 1
24
25 add esi, eax
26 loop next
27 _end:
28 mov eax, msg1
29 call sprintf
30 mov eax, msg2
31 call sprintf
32 mov eax, esi
33 call iprintf
34 call quit

```

Рис. 3.12: Код моей программы

Послек долгих попыток написать правильный код, я к этому пришла. Вот таким образом работает моя программа:(рис. 3.13).


```
dvmaksimova@dk3n55 ~/work/arch-pc/lab08 $ nasm -f elf nomer.asm
dvmaksimova@dk3n55 ~/work/arch-pc/lab08 $ ld -m elf_i386 -o nomer nomer.o
dvmaksimova@dk3n55 ~/work/arch-pc/lab08 $ ./nomer 2 2 2 2
Функция:  $f(x)=3x-1$ 
Результат: 20
dvmaksimova@dk3n55 ~/work/arch-pc/lab08 $ ./nomer 3 3
Функция:  $f(x)=3x-1$ 
Результат: 16
dvmaksimova@dk3n55 ~/work/arch-pc/lab08 $ ./nomer 5 7
Функция:  $f(x)=3x-1$ 
Результат: 34
dvmaksimova@dk3n55 ~/work/arch-pc/lab08 $ ./nomer 3 7
Функция:  $f(x)=3x-1$ 
Результат: 28
dvmaksimova@dk3n55 ~/work/arch-pc/lab08 $ █
```

Рис. 3.13: Работает успешно!

4 Выводы

Я приобрела навыки написания программ с использованием циклов и навыки обработки аргументов командной строки.