

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук
Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

ПО ЛАБОРАТОРНОЙ РАБОТЕ № 8

дисциплина: Архитектура компьютера

Студент: Мелкумян Арвин

Группа: НКАбд-04-23

МОСКВА

2023 г.

ПРОГРАММИРОВАНИЕ ЦИКЛА. ОБРАБОТКА АРГУМЕНТОВ КОМАНДНОЙ СТРОКИ.

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

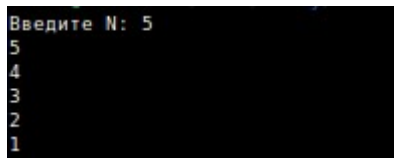
Ход работы.

Создадим программу для реализации простого цикла, исходный код которой показан на рисунке 1.

```
30 %include 'in_out.asm'
29
28 SECTION .data
27     msg1 db 'Введите N: ',0h
26
25 SECTION .bss
24     N: resb 10
23
22 SECTION .text
21     global _start
20
19 _start:
18     mov eax, msg1
17     call sprint
16
15     mov ecx, N
14     mov edx, 10
13     call sread
12
11     mov eax, N
10     call atoi
9     mov [N], eax
8
7     mov ecx, [N]
6 label:
5     mov [N], ecx
4     mov eax, [N]
3     call iprintLF
2     loop label
1
31     call quit
```

Рисунок 1 — Исходный код программы

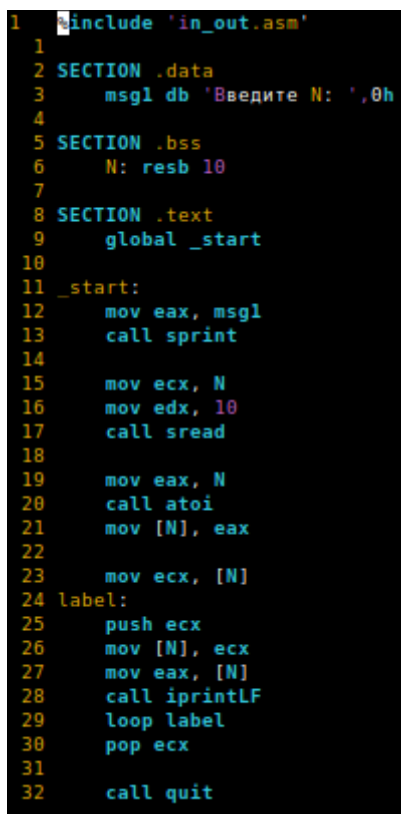
Данная программа реализует цикл, который выводит числа от заданного до единицы. Результат работы программы показан на рисунке 2.



```
Введите N: 5
5
4
3
2
1
```

Рисунок 2 — Результат работы программы

Подобная программа, но с сохранением значения в стеке показана на рисунке 2.



```
1 include 'in_out.asm'
2 SECTION .data
3     msg1 db 'Введите N: ',0h
4
5 SECTION .bss
6     N: resb 10
7
8 SECTION .text
9     global _start
10
11 _start:
12     mov eax, msg1
13     call sprint
14
15     mov ecx, N
16     mov edx, 10
17     call sread
18
19     mov eax, N
20     call atoi
21     mov [N], eax
22
23     mov ecx, [N]
24 label:
25     push ecx
26     mov [N], ecx
27     mov eax, [N]
28     call iprintLF
29     loop label
30     pop ecx
31
32     call quit
```

Рисунок 3 — Исходный код второй версии программы

Далее разработаем программу, которая выводит значения переданных ей аргументов командной строки. Исходный код показан на рисунке 4.

```

1  include 'in_out.asm'
2  SECTION .text
3      global _start
4
5  _start:
6      pop ecx
7      pop edx
8      sub ecx, 1
9
10 next:
11     cmp ecx, 0
12     jz _end
13
14     pop eax
15     call sprintfLF
16     loop next
17
18 _end:
19     call quit

```

Рисунок 4 — Исходный код программы

Результат работы данной программы показан на рисунке 5.

```

infer@Cameron:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc/labs/lab08$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3

```

Рисунок 5 — Результат работы программы

Как видно из рисунка, аргументом является любое значение, ограниченное пробелом либо одинарными кавычками.

Аналогично создадим программу, вычисляющую сумму аргументов командной строки. Ее код представлен на рисунке 6, а результат выполнения на рисунке 7.

```

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

```

Рисунок 6 — Исходный код программы

```

infer@Cameron:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc/labs/lab08$ ./lab8-3 1 2 3
Результат: 6

```

Рисунок 7 — Результат выполнения программы

В качестве самостоятельной работы создадим программу, которая складываем результаты вычисления функции аргументов, заданных в качестве аргументов командной строки. Согласно варианту целевая функция: $10 \cdot (x-1)$.

Исходный код программы показан на рисунке 8. Результат выполнения программы показан на рисунке 9.

```

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

```

Рисунок 8 — Исходный код программы

```

infer@Cameron:~/work/study/2023-2024/Архитектура компьютера/study_2023-2024_arh-pc/labs/lab08$ ./lab8-4 1 2 3 4
Результат: 60

```

Рисунок 9 — Результат выполнения программы

Выводы: В ходе лабораторной работы были приобретены навыки написания программ, использующих циклы и обработку аргументов командной строки в NASM.