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: Тимаков Макарий Владимирович

Группа: НПИбд 02-25

МОСКВА

2025г.

1.Цель работы

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

2. Выполнение лабораторной работы

2.1 Реализация циклов в NASM

Создаем каталог для программ ЛБ8, и в нем создаем файл (рис. Рисунок 2.1).

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

Рисунок 2.1: Создаем каталог с помощью команды mkdir и файл с помощью команды touch

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 8.1 (рис. Рисунок 2.2).

```
%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
```

Рисунок 2.2: Заполняем файл

Создаем исполняемый файл и запускаем его (рис. Рисунок 2.3).

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

Рисунок 2.3: Запускаем файл и проверяем его работу

Снова открываем файл для редактирования и изменяем его, добавив изменение значения регистра в цикле (рис. Рисунок 2.4)

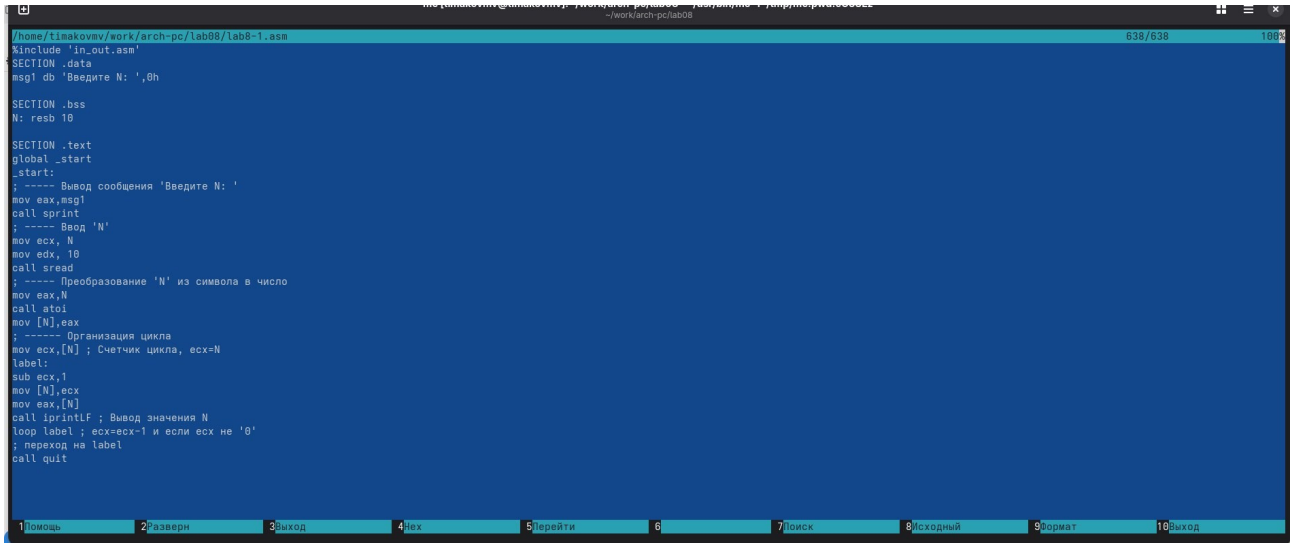

Рисунок 2.4: Изменяем файл

Создаем исполняемый файл и запускаем его (рис. Рисунок 2.5).

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

Рисунок 2.5: Запускаем файл и смотрим на его работу

Регистр ecx принимает значения 9,7,5,3,1(на вход подается число 10, в цикле label данный регистр уменьшается на 2 командой sub и loop).
Число проходов цикла не соответсвует числу N, так как уменьшается на 2.

Снова открываем файл для редактирования и изменяем его, чтобы все корректно работало (рис. Рисунок 2.6).

Рисунок 2.6: Редактируем файл

Создаем исполняемый файл и запускаем его (рис. Рисунок 2.7).

Рисунок 2.7: Проверяем, сошелся ли наш вывод с данным в условии выводом

В данном случае число проходов цикла равна числу N

2.2 Обработка аргументов командной строки.

Создаем новый файл (рис. Рисунок 2.8).

```
timakovmv@timakovmv:~/work/arch-pc/lab08$ touch lab8-2
timakovmv@timakovmv:~/work/arch-pc/lab08$
```

Рисунок 2.8: Создаем файл командой touch

Открываем файл в Midnight Commander и заполняем его в соответствии с листингом 8.2 (рис. Рисунок 2.9).

```
/home/timakovmv/work/arch-pc/lab08/lab8-2.asm                    933/933  100%
%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 sprintLF ; вызываем функцию печати
loop next ; переход к обработке следующего
; аргумента (переход на метку next)
_end:
call quit

 1Помощь  2Разверн  3Выход  4Hex  5Перейти  6  7Поиск  8Исходный  9Формат  10Выход
```

Рисунок 2.9: Заполняем файл

Создаем исполняемый файл и проверяем его работу, указав аргументы (рис. Рисунок 2.10).

```
timakovmv@timakovmv:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
timakovmv@timakovmv:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
timakovmv@timakovmv:~/work/arch-pc/lab08$ ./lab8-2
timakovmv@timakovmv:~/work/arch-pc/lab08$ ./lab8-2 1 2 '3'
1
2
3
timakovmv@timakovmv:~/work/arch-pc/lab08$ ~
```

Рисунок 2.10: Смотрим на работу программ

Програмой было обработано 3 аргумента.

Создаем новый файл lab8-3.asm (рис. Рисунок 2.11).

```
timakovmv@timakovmv:~/work/arch-pc/lab08$ touch lab8-3.asm
timakovmv@timakovmv:~/work/arch-pc/lab08$
```

Рисунок 2.11: Создаем файл командой touch

Открываем файл и заполняем его в соответствии с листингом 8.3 (рис. Рисунок 2.12).

```
/home/timakovmv/work/arch-pc/lab08/lab8-3.asm                    1414/1414
%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
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

1Помощь 2Разверн 3Выход 4Hex 5Перейти 6 7Поиск 8Исходный 9Формат 10Выход
```

Рисунок 2.12: Заполняем файл

Создаём исполняемый файл и запускаем его, указав аргументы (рис. Рисунок 2.13).

```
timakovmv@timakovmv:~/work/arch-pc/lab08$  nasm -f elf lab8-3.asm
timakovmv@timakovmv:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.
timakovmv@timakovmv:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
timakovmv@timakovmv:~/work/arch-pc/lab08$
```

Рисунок 2.13: Смотрим на работу программы

Снова открываем файл для редактирования и изменяем его, чтобы вычислялосьпроизведение вводимых значений (рис. Рисунок 2.14).

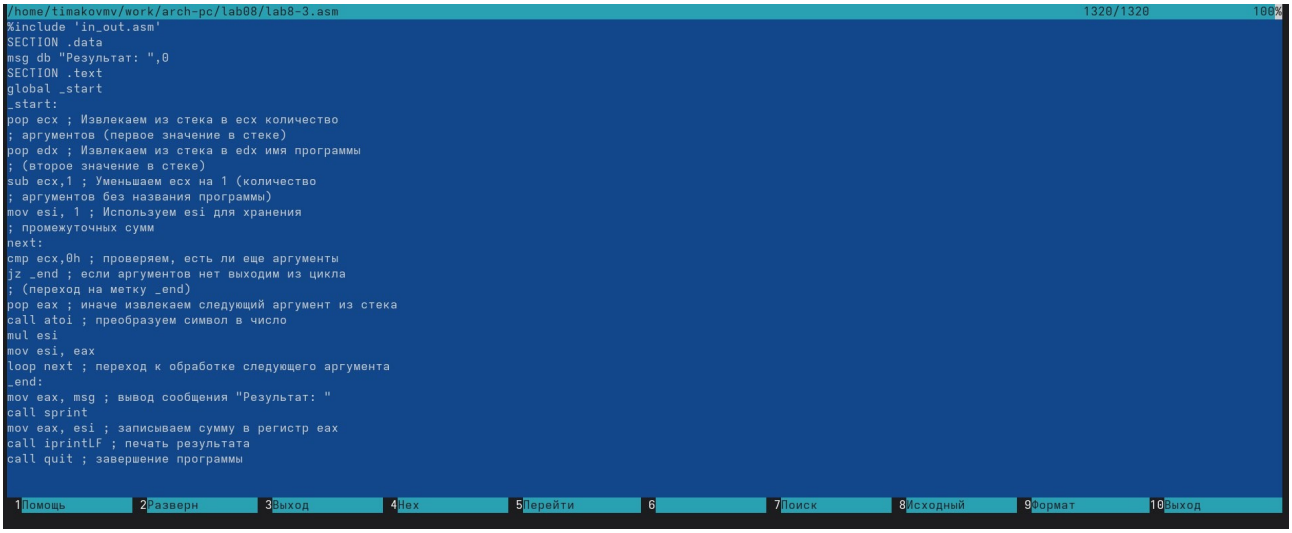

Рисунок 2.14: Изменяем файл

Создаём исполняемый файл и запускаем его, указав аргументы (рис. Рисунок 2.15).

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

Рисунок 2.15: Проверяем работу файла(работает правильно)

3. Задание для самостоятельной работы

ВАРИАНТ 19

1. Напишите программу, которая находит сумму значений функции f(x) для x = x1, x2, …, xn, т.е. программа должна выводить значение f(x1) + f(x2) + … + f(xn). Значения xi передаются как аргументы. Вид функции f(x) выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы № 7. Создайте исполняемый файл и проверьте его работу на нескольких наборах x = x1, x2, …, x2.

Создаем новый файл (рис. Рисунок 3.1).

```
timakovmv@timakovmv:~/work/arch-pc/lab08$ touch lab8-4.asm
timakovmv@timakovmv:~/work/arch-pc/lab08$
```

Рисунок 3.1: Создаем файл командой touch

Открываем его и пишем программу, которая выведет сумму значений, получившихся после решения выражения 8x-3 (рис. Рисунок 3.2)

Рисунок 3.2: Пишем программу

Транслируем файл и смотрим на работу программы (рис. Рисунок 3.3).

```
timakovmv@timakovmv:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
timakovmv@timakovmv:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
timakovmv@timakovmv:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
timakovmv@timakovmv:~/work/arch-pc/lab08$ ./lab8-4 1 2 3
Результат: 39
timakovmv@timakovmv:~/work/arch-pc/lab08$
```

Рисунок 3.3: Смотрим на рабботу программы при x1=1 x2=2 x2=3(всё верно)

4.Выводы

Мы научились решать программы с использованием циклов и обработкой аргументов командной строки.