Отчет по лабораторной работе №8

Дисциплина: Архитектура компьютера

Чипурной Михаил Евгеньевич

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Изучить работу 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).

```
mechipurnoyj@fedora:~$ mkdir ~/work/arch-pc/lab08
mechipurnoyj@fedora:~$ cd ~/work/arch-pc/lab08
mechipurnoyj@fedora:~/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).

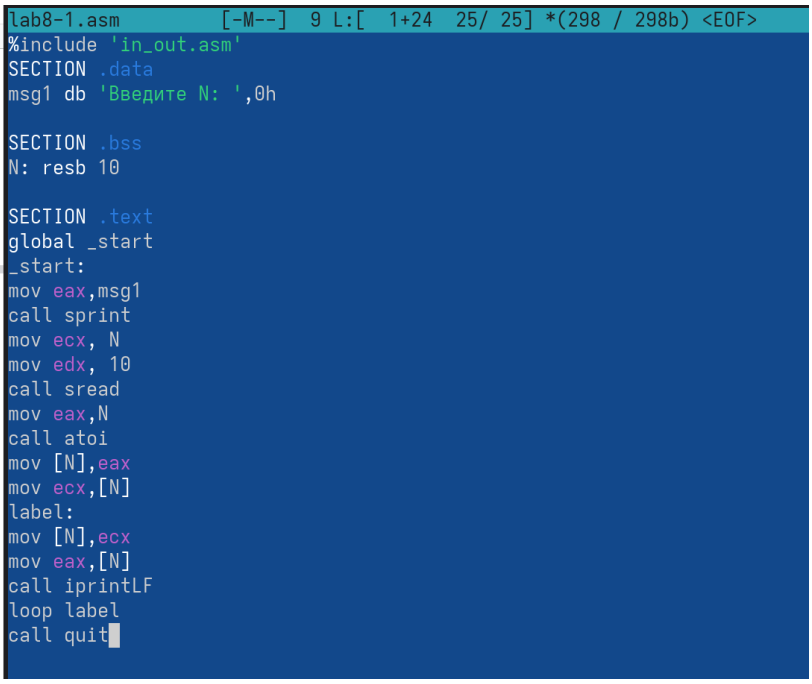

Рисунок 2.2: Заполняем файл

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

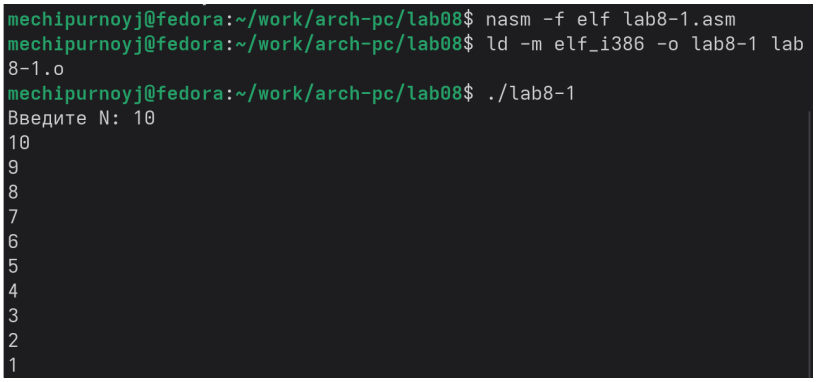

Рисунок 2.3: Запускаем файл и проверяем его работу

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

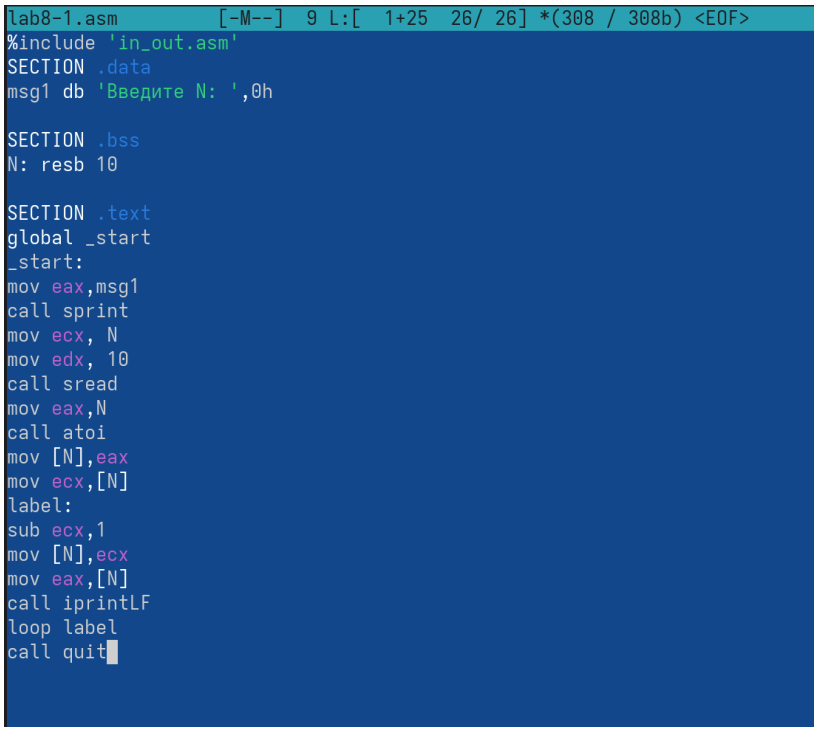

Рисунок 2.4: Изменяем файл

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

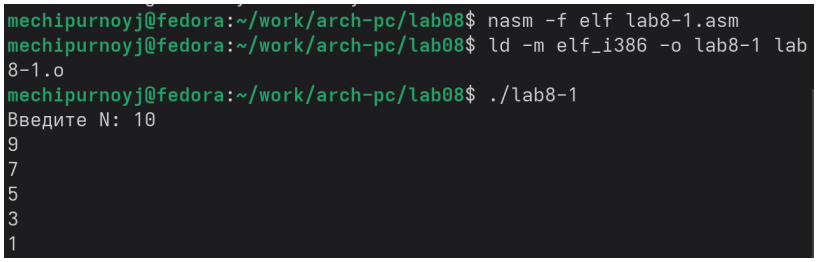

Рисунок 2.5: Запускаем файл и смотрим на его работу

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).

```
lab8-1.asm          [-M--]  9 L:[  1+27  28/ 28] *(325 / 325b) <EOF>
%include 'in_out.asm'
SECTION .data
msg1 db 'Введите N: ',0h

SECTION .bss
N: resb 10

SECTION .text
global _start
_start:
mov eax,msg1
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
call quit
```

Рисунок 2.6: Редактируем файл

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

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

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).

```
mechipurnoyj@fedora:~/work/arch-pc/lab08$ touch lab8-2.asm
mechipurnoyj@fedora:~/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).

```
lab8-2.asm          [-M--]  0 L:[  1+19  20/ 20] *(198 / 198b) <EOF>
%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
```

Рисунок 2.9: Заполняем файл

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

```
mechipurnoyj@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
mechipurnoyj@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab
8-2.o
mechipurnoyj@fedora:~/work/arch-pc/lab08$ ./lab8-2 1 2 '3'
1
2
3
mechipurnoyj@fedora:~/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).

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

```
lab8-3.asm          [-M--] 13 L:[  1+27  28/ 28] *(341 / 341b) <EOF>
%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
```

Рисунок 2.12: Заполняем файл

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

```
mechipurnoyj@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
mechipurnoyj@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab
8-3.o
mechipurnoyj@fedora:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
mechipurnoyj@fedora:~/work/arch-pc/lab08$
```

Рисунок 2.13: Смотрим на работу программы

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

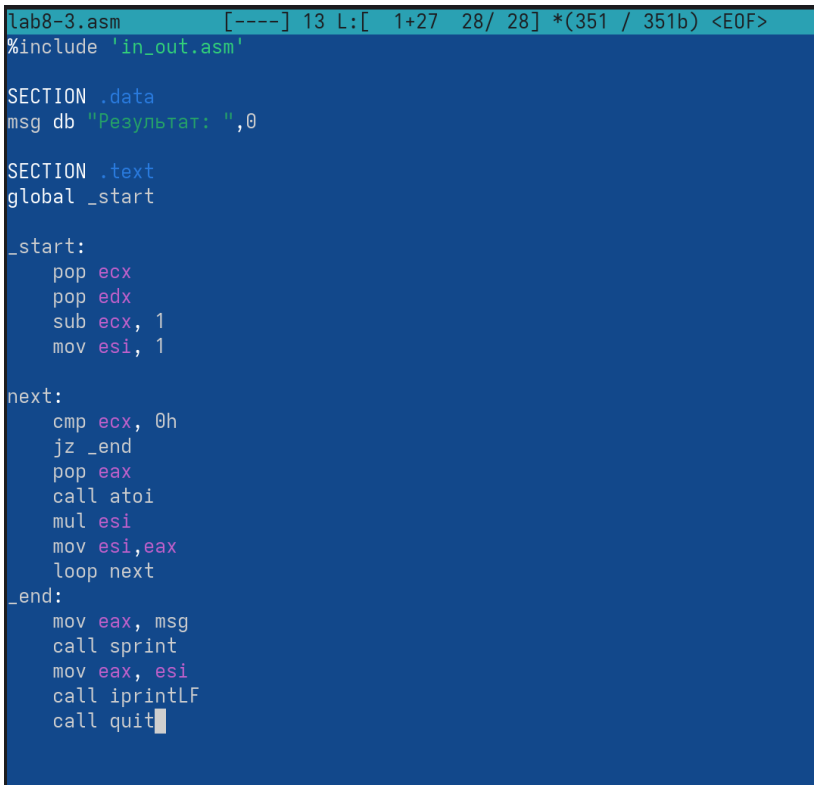

Рисунок 2.14: Изменяем файл

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

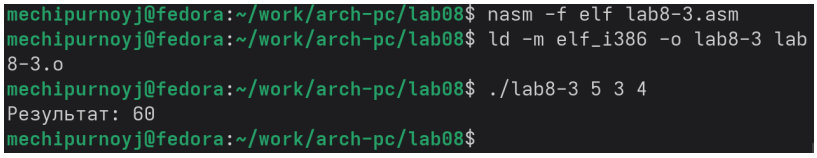

Рисунок 2.15: Проверяем работу файла(работает правильно)

2.3 Задание для самостоятельной работы

ВАРИАНТ-17

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 2.16).

Рисунок 2.16: Создаем файл командой touch

Открываем его и пишем программу, которая выведет сумму значений, получившихся после решения выражения 10(x-1) (рис. Рисунок 2.17).

```
lab8-4.asm          [-M--] 13 L:[  1+32  33/ 33] *(400 / 400b) <EOF>
%include "in_out.asm"

SECTION .data
msg db "Результат: ",0

SECTION .bss
prm: resb 80

SECTION .text
global _start

_start:
    pop ecx
    pop edx
    sub ecx, 1
    mov esi, 10

next:
    cmp ecx, 0h
    jz _end
    pop eax
    call atoi
    sub eax, 1
    mul esi
    add [prm], eax
    loop next

_end:
    mov eax, msg
    call sprint
    mov eax, [prm]
    call iprintLF
    call quit
```

Рисунок 2.17: Пишем программу

Транслируем файл и смотрим на работу программы (рис. Рисунок 2.18).

```
mechipurnoyj@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
mechipurnoyj@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab
8-4.o
mechipurnoyj@fedora:~/work/arch-pc/lab08$ ./lab8-4 5 3 4
Результат: 90
mechipurnoyj@fedora:~/work/arch-pc/lab08$
```

Рисунок 2.18: Смотрим на рабботу программы при x1=5 x2=3 x1=4(всё верно)

Транслируем файл и смотрим на работу программы (рис. Рисунок 2.19).

```
mechipurnoyj@fedora:~/work/arch-pc/lab08$ ./lab8-4 6 7 2
Результат: 120
```

Рисунок 2.19: Смотрим на рабботу программы при x1=6 x2=7 x1=2(всё верно)

3 Выводы

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