

Лабораторная работа №8

Программирование цикла. Обработка аргументов командной строки

Акунаева Антонина Эрдниевна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
3.1	Реализация циклов в NASM	7
3.2	Обработка аргументов командной строки	11
4	Описание результатов выполнения заданий для самостоятельной работы	17
5	Выводы	19

Список иллюстраций

3.1	Использование команд mkdir и touch	7
3.2	Mscedit: листинг 8.1 в файле lab8-1.asm	8
3.3	Создание и запуск исполняемого файла lab8-1	9
3.4	Mscedit: изменения в файле lab8-1.asm: некорректная работа	9
3.5	Создание и запуск исполняемого файла lab8-1: некорректная работа	10
3.6	Mscedit: изменения в файле lab8-1.asm: добавление стека	10
3.7	Создание и запуск исполняемого файла lab8-1: добавление стека .	11
3.8	Создание lab8-2.asm при помощи touch	11
3.9	Mscedit: листинг 8.2 в файле lab8-2.asm	12
3.10	Создание и запуск исполняемого файла lab8-2	13
3.11	Создание lab8-3.asm при помощи touch	13
3.12	Mscedit: листинг 8.3 в файле lab8-3.asm	14
3.13	Создание и запуск исполняемого файла lab8-3	14
3.14	Mscedit: файл lab8-3.asm: произведение аргументов	15
3.15	Создание и запуск исполняемого файла lab8-3: произведение аргу- ментов	15
4.1	Создание lab8-4.asm при помощи touch	17
4.2	Mscedit: lab8-4.asm	18
4.3	Создание и запуск исполняемого файла lab8-4	18

Список таблиц

1 Цель работы

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

2 Задание

Научиться реализовывать циклы в NASM.

Научиться обрабатывать аргументы командной строки в NASM.

3 Выполнение лабораторной работы

3.1 Реализация циклов в NASM

3.1.1. Создайте каталог для программ лабораторной работы №6, перейдите в него и создайте файл lab8-1.asm.



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

Рис. 3.1: Использование команд mkdir и touch

Создадим каталог lab08 в рабочем каталоге при помощи mkdir, перейдём в него с cd. В новом каталоге создадим NASM-файл lab8-1.asm при помощи touch.

Внимательно изучите текст программы (Листинг 8.1). Введите в файл lab8-1.asm текст программы из листинга 8.1. Создайте исполняемый файл и проверьте его работу.

```

lab8-1.asm      [----] 16 L:[ 1+13 14/ 32] *(424 / 917b) 0[*][X]
;-----
; Программа вывода значений регистра 'ecx'
;-----
#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
1По~щъ 2Со~ан 3Блок 4За~на 5Копия 6Пе~ть 7Поиск 8Уд~ть 9Ме~МС

```

Рис. 3.2: Mscedit: листинг 8.1 в файле lab8-1.asm


```

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

```

Рис. 3.3: Создание и запуск исполняемого файла lab8-1

Изучим текст листинга 8.1 - в нём описан текст программы для вывода значений регистра 'ecx', используемого в кач-ве счётчика для инструкции цикла loop.

Откроем файл .asm в текстовом редакторе mcedit. Скопируем текст листинга в файл lab8-1.asm и создадим исполняемый файл. Запустим его и получим построчный отсчёт от введённого нами числа до 1.

Измените текст программы добавив изменение значение регистра ecx в цикле.

```

label:
    sub ecx,1 ; `ecx=ecx-1`
    mov [N],ecx
    mov eax,[N]
    call iprintLF ; Вывод значения `N`
    loop label ; `ecx=ecx-1` и если `ecx` не '0'
; переход на `label`

```

Рис. 3.4: Mcedit: изменения в файле lab8-1.asm: некорректная работа

Снова откроем в mcedit lab8-1.asm и отредактируем строки в теле label. Мы добавляем одну строку sub ecx,1 в цикл, который дополнительно изменяет ecx.

Создайте исполняемый файл и проверьте его работу. Какие значения принимает регистр ecx в цикле? Соответствует ли число проходов цикла значению N

введенному с клавиатуры?

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

Рис. 3.5: Создание и запуск исполняемого файла lab8-1: некорректная работа

Оттранслируем и создадим исполняемый файл, запустим его и получим в результате не числа от 1 до 10, а только нечётные. Это происходит из-за добавленной строки, т.к. дополнительно к действию цикла над регистром `ecx` отнимается ещё 1. Программа работает, однако количество проходов цикла не совпадает значению введённого `N`.

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

```
label:
    push ecx
    sub ecx,1 ; `ecx=ecx-1`
    mov [N],ecx
    mov eax,[N]
    call iprintLF ; Вывод значения `N`
    pop ecx
    loop label ; `ecx=ecx-1` и если `ecx` не '0'
; переход на `label`
```

Рис. 3.6: Mscedit: изменения в файле lab8-1.asm: добавление стека

Внесём в текст программы ещё одни изменения - добавим стек. Другими словами, будем добавлять значение регистра `ecx` в стек, специальную структуру данных для хранения, а потом извлекать `ecx` из стека.

Создайте исполняемый файл и проверьте его работу. Соответствует ли в данном случае число проходов цикла значению N введённому с клавиатуры?

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

Рис. 3.7: Создание и запуск исполняемого файла lab8-1: добавление стека

Запустим новый исполняемый файл и получим уже 10 проходов цикла, как и было указано в числе N, однако счёт начинается с 9 и заканчивается 0, вместо 10 и 1, т.к. в цикл передаётся сохранённое значение esx 10б из него отнимается 1 и печатается на экран и т.д.

3.2 Обработка аргументов командной строки

3.2.1. Внимательно изучите текст программы (Листинг 8.2). Создайте файл lab8-2.asm в каталоге ~/work/arch-pc/lab08 и введите в него текст программы из листинга 8.2.

```
aeakunaeva@fedora:~/work/arch-pc/lab08$ touch lab8-2.asm
aeakunaeva@fedora:~/work/arch-pc/lab08$ ls
in_out.asm  lab8-1  lab8-1.asm  lab8-1.o  lab8-2.asm
aeakunaeva@fedora:~/work/arch-pc/lab08$
```

Рис. 3.8: Создание lab8-2.asm при помощи touch

```

lab8-2.asm      [----] 13 L:[ 1+24 25/ 25] *(1236/123
;-----
; Обработка аргументов командной строки
;-----
%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

```

Рис. 3.9: Mcedit: листинг 8.2 в файле lab8-2.asm

Ознакомимся с листингом 8.2 - программа, выводящая на экран аргументы командной строки. Создадим NASM-файл lab8-2.asm при помощи touch в текущей директории. Скопируем в него текст листинга 8.2.

Создайте исполняемый файл и запустите его, указав аргументы. Сколько аргументов было обработано программой?

```

aeakunaeva@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
aeakunaeva@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
aeakunaeva@fedora:~/work/arch-pc/lab08$ ./lab8-2
aeakunaeva@fedora:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент 2
'аргумент 3'
аргумент1
аргумент
2
аргумент 3
aeakunaeva@fedora:~/work/arch-pc/lab08$

```

Рис. 3.10: Создание и запуск исполняемого файла lab8-2

Запустим созданный исполняемый файл и проверим его работу. При вводе различных аргументов через пробел выводятся 4 аргумента (те, которые были указаны через пробел, а не логические).

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

```

aeakunaeva@fedora:~/work/arch-pc/lab08$ touch lab8-3.asm
aeakunaeva@fedora:~/work/arch-pc/lab08$ ls
in_out.asm  lab8-1.asm  lab8-2      lab8-2.o
lab8-1      lab8-1.o    lab8-2.asm  lab8-3.asm
aeakunaeva@fedora:~/work/arch-pc/lab08$

```

Рис. 3.11: Создание lab8-3.asm при помощи touch

```

lab8-3.asm      [----]  4 L:[  1+31  32/ 32] *(1470/1521b) 0[*][X]
#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

```

Рис. 3.12: Mscedit: листинг 8.3 в файле lab8-3.asm

Создадим NASM-файл lab8-3.asm при помощи touch в текущей директории. Скопируем в него текст листинга 8.3.

Создайте исполняемый файл и запустите его, указав аргументы.

```

aeakunaeva@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
aeakunaeva@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
aeakunaeva@fedora:~/work/arch-pc/lab08$ ./lab8-3 34 5 9 0
Результат: 48
aeakunaeva@fedora:~/work/arch-pc/lab08$

```

Рис. 3.13: Создание и запуск исполняемого файла lab8-3

Запустим созданный исполняемый файл и проверим его работу. Всё выполня-

нётся корректно: программа выводит сумму введённых аргументов.

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

```
lab8-3.asm [----] 13 L: [ 1+15 16/ 32] *(552 /1577b) 0[*][X]
%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,1 ; Используем `esi` для хранения
<-----> ; промежуточных произведений
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
    mul esi ; добавляем к промежуточному произведению
    mov esi,eax ; 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 произведение в регистр `eax`
    call iprintLF ; печать результата
    call quit ; завершение программы
```

Рис. 3.14: Mcedit: файл lab8-3.asm: произведение аргументов

```
aeakunaeva@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
aeakunaeva@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
aeakunaeva@fedora:~/work/arch-pc/lab08$ ./lab8-3 6 2 3
Результат: 36
aeakunaeva@fedora:~/work/arch-pc/lab08$
```

Рис. 3.15: Создание и запуск исполняемого файла lab8-3: произведение аргументов

Изменим строку `add esi,eah` на `mul esi`(чтобы $eah * esi$) и добавим строку `mov esi,eah`, чтобы сохранить в `esi` значение промежуточного произведения и изменим начальное значение `esi` на 1 вместо 0. Запустим созданный исполняемый файл и проверим его работу. Всё выполняется корректно: программа выводит произведение введенных аргументов.

4 Описание результатов выполнения заданий для самостоятельной работы

4.1. Напишите программу, которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, \dots, x_n$, т.е. программа должна выводить значение $f(x_1) + f(x_2) + \dots + f(x_n)$. Значения x_i передаются как аргументы. Вид функции $f(x)$ выбрать из таблицы 8.1 вариантов заданий в соответствии с вариантом, полученным при выполнении лабораторной работы №6. Создайте исполняемый файл и проверьте его работу на нескольких наборах $x = x_1, x_2, \dots, x_n$.

Вариант 13. $f(x)=12x-7$

```
aeakunaeva@fedora:~/work/arch-pc/lab08$ touch lab8-4.asm
aeakunaeva@fedora:~/work/arch-pc/lab08$ ls
in_out.asm  lab8-1.asm  lab8-2      lab8-2.o  lab8-3.asm  lab8-4.asm
lab8-1      lab8-1.o    lab8-2.asm  lab8-3    lab8-3.o
```

Рис. 4.1: Создание lab8-4.asm при помощи touch

```

lab8-4.asm [----] 27 L: [ 6+19 25/ 35] *(1082/1610b) 0010 0x00A [*][X]
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
    mov ebx,12 ; ebx = 12
    mul ebx ; eax (x) * 12
    sub eax,7 ; eax - 7
    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 "f(x1)+f(x2)+...+f(n) = "
    call sprint
    mov eax, esi ; записываем сумму в регистр 'eax'
    call iprintLF ; печать результата
1Помощь 2Сохран 3Блок 4Замена 5Копия 6Пер-ить 7Поиск 8Удалить 9МенюМС 10Выход

```

Рис. 4.2: Mcedit: lab8-4.asm

Создадим NASM-файл lab8-4.asm при помощи touch в текущей директории. Запишем в него текст программы, соответствующей условию со значениями и функцией из варианта 13.

```

aeakunaeva@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
aeakunaeva@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
aeakunaeva@fedora:~/work/arch-pc/lab08$ ./lab8-4
f(x1)+f(x2)+...+f(xn) = 0
aeakunaeva@fedora:~/work/arch-pc/lab08$ ./lab8-4 1 2 3
f(x1)+f(x2)+...+f(xn) = 51
aeakunaeva@fedora:~/work/arch-pc/lab08$ ./lab8-4 4 0 13
f(x1)+f(x2)+...+f(xn) = 183
aeakunaeva@fedora:~/work/arch-pc/lab08$

```

Рис. 4.3: Создание и запуск исполняемого файла lab8-4

Оттранслируем объектный файл и создадим исполняемый, запустим его. В результате выводится сумма значений функции $f(x) = 12x - 7$ при разных введенных аргументах.

5 Выводы

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