

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ

по лабораторной работе № 9

дисциплина: Архитектура компьютера

Студент: Ибрагимов Мурад

Группа: НПИбд-01-22

МОСКВА

2022 г.

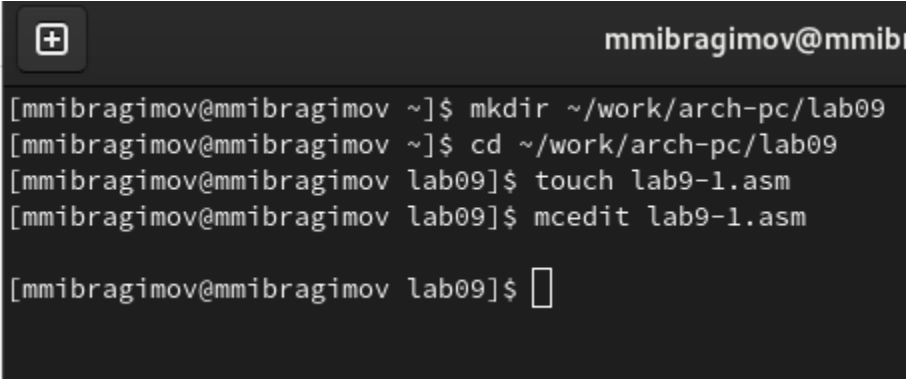
Цель работы:

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

Порядок выполнения лабораторной работы:

Реализация циклов в NASM.

Создадим каталог для программ лабораторной работы №9, перейдем в него и создадим нужный файл (рис. 1).



```
mmibragimov@mmibragimov ~$ mkdir ~/work/arch-pc/lab09
mmibragimov@mmibragimov ~$ cd ~/work/arch-pc/lab09
mmibragimov@mmibragimov lab09$ touch lab9-1.asm
mmibragimov@mmibragimov lab09$ mcedit lab9-1.asm

mmibragimov@mmibragimov lab09$ █
```

рис. 1. Создание файла lab9-1.asm

При реализации циклов в NASM с использованием инструкции loop необходимо помнить о том, что эта инструкция использует регистр ecx в качестве счетчика и на каждом шаге уменьшает его значение на единицу. В качестве примера рассмотрим программу, которая выводит значение регистра ecx (рис. 2).

```

lab9-1.asm      [-M--]  6 L: [ 1+21 22/ 28] *(472 /
#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

```

рис. 2. Текст программы lab9-1

Создадим исполняемый файл и проверим его работу (рис. 3).

```

[mmibragimov@mmibragimov lab09]$ mcedit lab9-1.asm

[mmibragimov@mmibragimov lab09]$ nasm -f elf lab9-1.asm
[mmibragimov@mmibragimov lab09]$ ld -m elf_i386 lab9-1.o -o lab9-1
[mmibragimov@mmibragimov lab09]$ ./lab9-1
Введите N: 4
3
1
[mmibragimov@mmibragimov lab09]$ 

```

рис. 3. Результат работы программы lab9-1

Данный пример показывает, что использование регистра `ecx` в теле цикла `loop` может привести к некорректной работе программы. Изменим текст программы добавив изменение значения регистра `ecx` в цикле по следующему примеру (рис. 4).

```
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
sub 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`
call quit
```

рис. 4. Пример изменения части программы lab9-1

Создадим исполняемый файл и проверим его работу (рис. 5).

```
[mmibragimov@mmibragimov lab09]$
[mmibragimov@mmibragimov lab09]$
[mmibragimov@mmibragimov lab09]$ nasm -f elf lab9-1.asm
[mmibragimov@mmibragimov lab09]$ ld -m elf_i386 lab9-1.o -o lab9-1
[mmibragimov@mmibragimov lab09]$ ./lab9-1
Введите N: 3
3
2
1
[mmibragimov@mmibragimov lab09]$
```

рис. 5. Результат работы измененной программы lab9-1

Как видим, все работает. Регистр `ecx` принимает все значения от `N` до 1 включительно, что соответствует числу проходов цикла, введенному с клавиатуры.

Для использования регистра `ecx` в цикле и сохранения корректности работы программы можно использовать стек. Внесем изменения в текст программы по примеру, добавив команды `push` и `pop` (добавления в стек и

извлечения из стека) для сохранения значения счетчика цикла loop (рис. 6).

```
mov [N],ecx
; ----- Организация цикла
mov ecx,[N] ; Счетчик цикла, `ecx=N`
label:
push ecx
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
```

рис. 6. Внесение команд push и pop в текст программы lab9-1

Создадим исполняемый файл и проверим его работу (рис. 7).

```
1
[mmibragimov@mmibragimov lab09]$ mcedit lab9-1.asm

[mmibragimov@mmibragimov lab09]$ nasm -f elf lab9-1.asm
[mmibragimov@mmibragimov lab09]$ ld -m elf_i386 lab9-1.o -o lab9-1
[mmibragimov@mmibragimov lab09]$ ./lab9-1
Введите N: 3
3
2
1
[mmibragimov@mmibragimov lab09]$
```

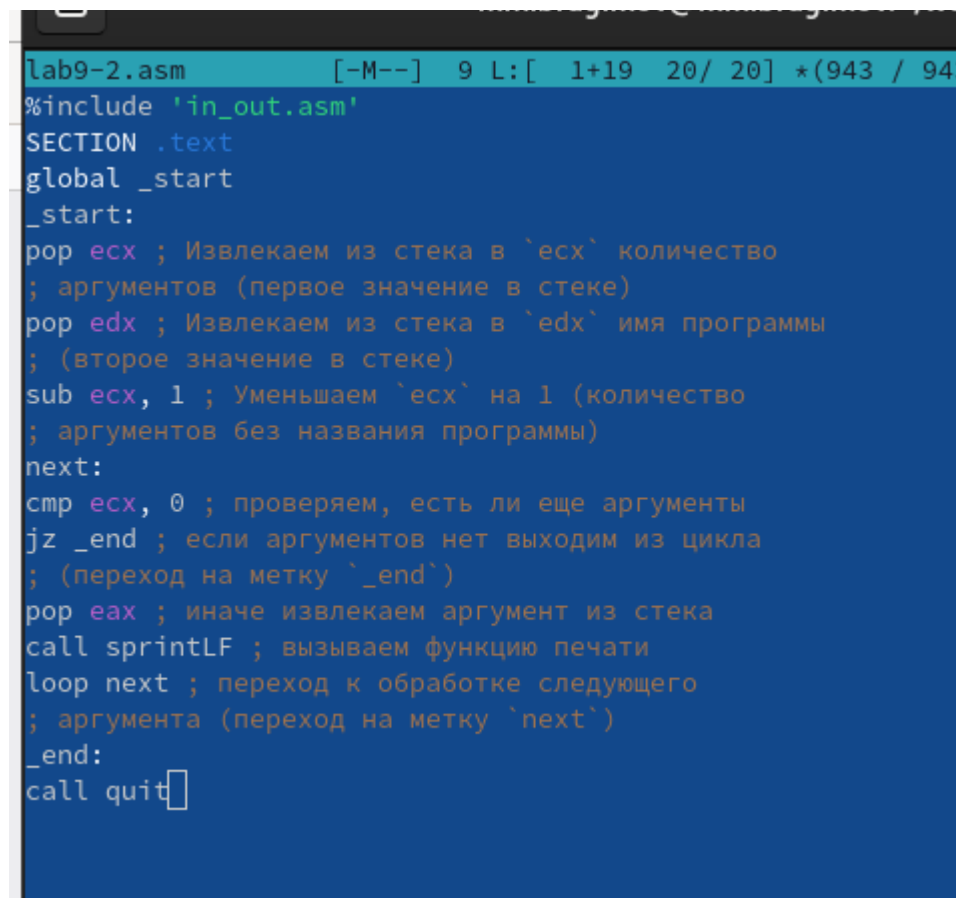
рис. 7. Результат работы измененной программы lab9-1

Как видим, программа работает корректно, число проходов цикла соответствует значению N, введенному с клавиатуры.

Обработка аргументов командной строки.

При разработке программ иногда встает необходимость указывать аргументы, которые будут использоваться в программе, непосредственно из командной строки при запуске программы. При запуске программы в NASM аргументы командной строки загружаются в стек в обратном порядке, кроме того в стек записывается имя программы и общее

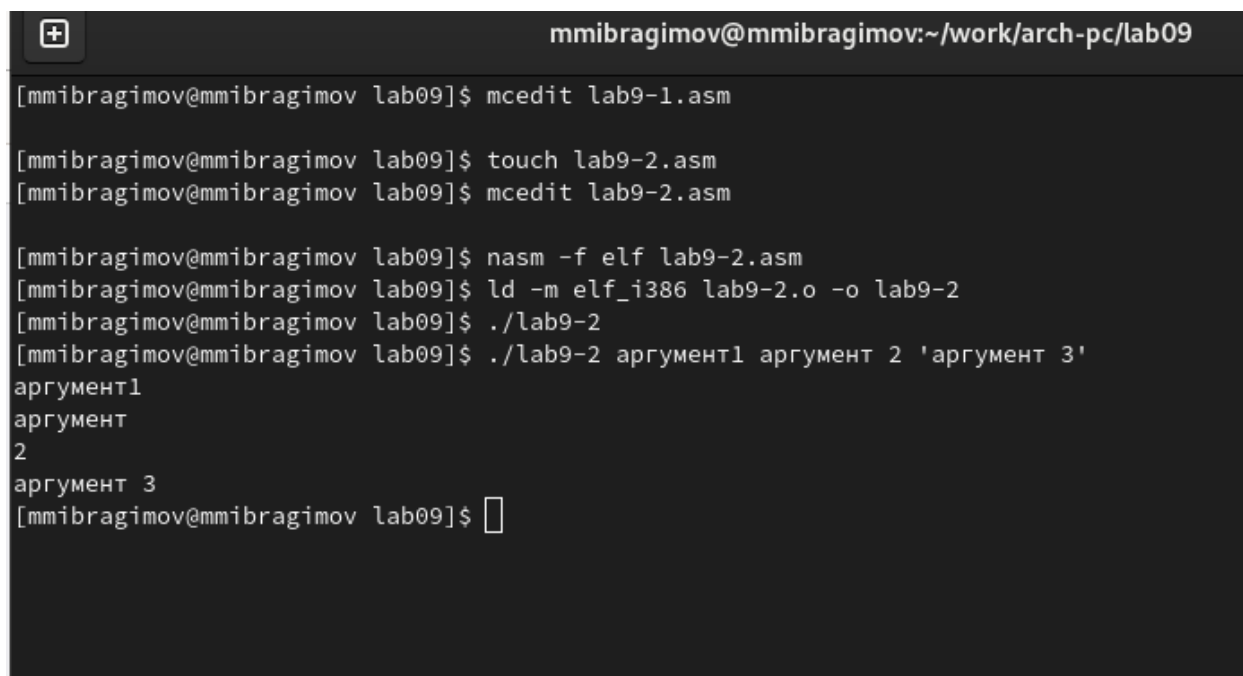
количество аргументов. Последние два элемента стека для программы, скомпилированной NASM, – это всегда имя программы и количество переданных аргументов. Таким образом, для того чтобы использовать аргументы в программе, их просто нужно извлечь из стека. Обработку аргументов нужно проводить в цикле. Т.е. сначала нужно извлечь из стека количество аргументов, а затем циклично для каждого аргумента выполнить логику программы. В качестве примера рассмотрим программу, которая выводит на экран аргументы командной строки (рис. 8). Создадим в каталоге лабораторной работы №9 файл lab9-2 и введем текст из рис. 8.



```
lab9-2.asm      [-M--]  9 L: [ 1+19 20/ 20] *(943 / 94)
%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. 8. Текст программы lab9-2

Затем создадим исполняемый файл и запустим программу, указав следующие аргументы (рис. 9).

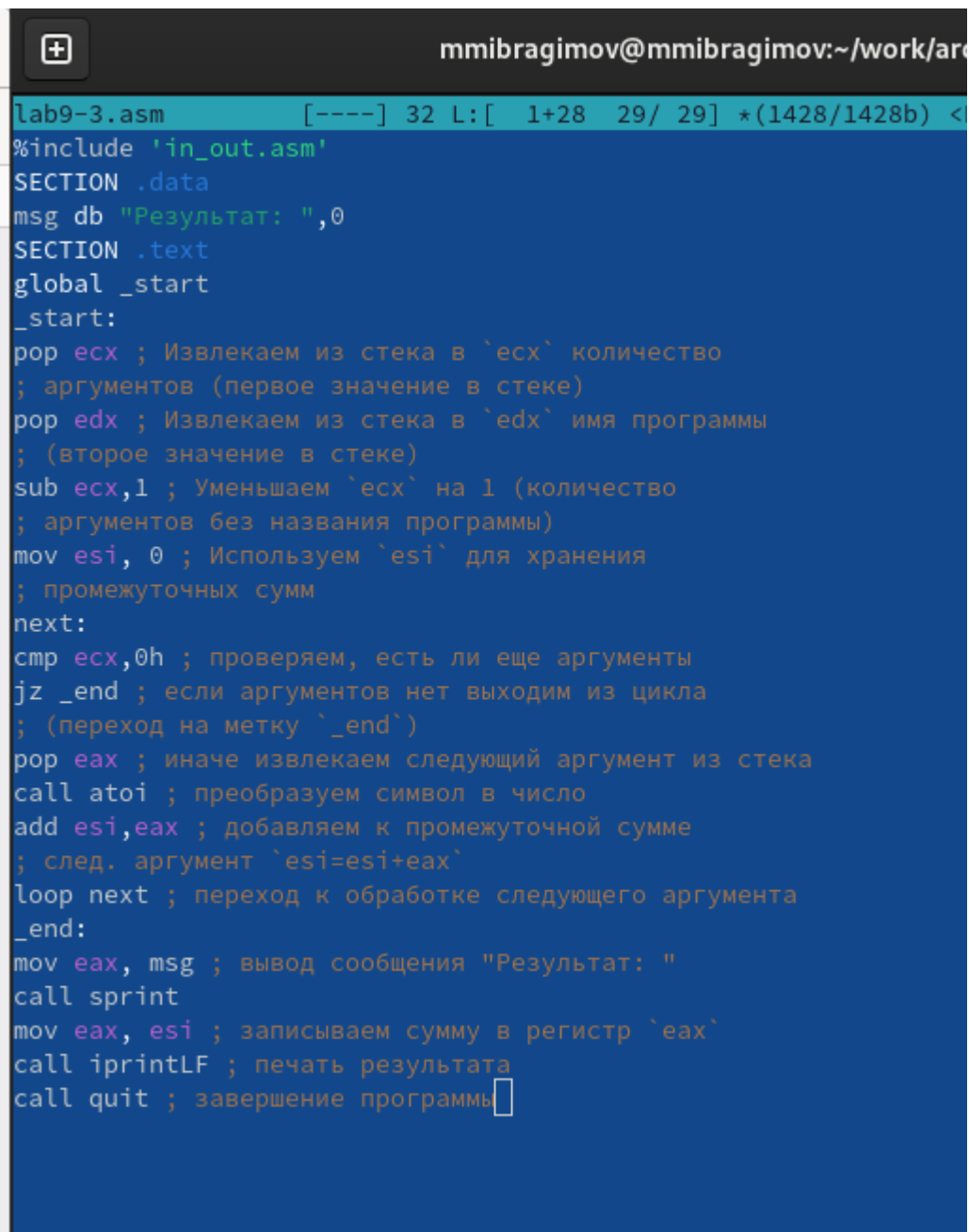
A terminal window with a dark background and light text. The title bar at the top shows a plus icon and the text 'mmibragimov@mmibragimov:~/work/arch-pc/lab09'. The terminal content shows a series of commands and their outputs. The commands are: 'mcedit lab9-1.asm', 'touch lab9-2.asm', 'mcedit lab9-2.asm', 'nasm -f elf lab9-2.asm', 'ld -m elf_i386 lab9-2.o -o lab9-2', './lab9-2', and './lab9-2 аргумент1 аргумент 2 'аргумент 3''. The outputs are: 'аргумент1', 'аргумент', '2', and 'аргумент 3'. The prompt '[mmibragimov@mmibragimov lab09]\$' is visible at the end of the last command line.

```
mmibragimov@mmibragimov:~/work/arch-pc/lab09
[mmibragimov@mmibragimov lab09]$ mcedit lab9-1.asm
[mmibragimov@mmibragimov lab09]$ touch lab9-2.asm
[mmibragimov@mmibragimov lab09]$ mcedit lab9-2.asm
[mmibragimov@mmibragimov lab09]$ nasm -f elf lab9-2.asm
[mmibragimov@mmibragimov lab09]$ ld -m elf_i386 lab9-2.o -o lab9-2
[mmibragimov@mmibragimov lab09]$ ./lab9-2
[mmibragimov@mmibragimov lab09]$ ./lab9-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
[mmibragimov@mmibragimov lab09]$
```

рис. 9. Результат работы программы lab9-2

Как видим, программа восприняла “аргумент” и “2” как отдельные аргументы, в то время как ‘аргумент 3’ как один. Соответственно программой было обработано 4 аргумента.

Рассмотрим еще один пример программы которая выводит сумму чисел, которые передаются в программу как аргументы. Создадим файл lab9-3.asm в том же каталоге и введем в него следующий текст программы (рис. 10).



```
lab9-3.asm [----] 32 L: [ 1+28 29/ 29] *(1428/1428b) <
#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. 10. Текст программы lab9-3

Затем создадим исполняемый файл и запустим его, указав аргументы (рис. 11).

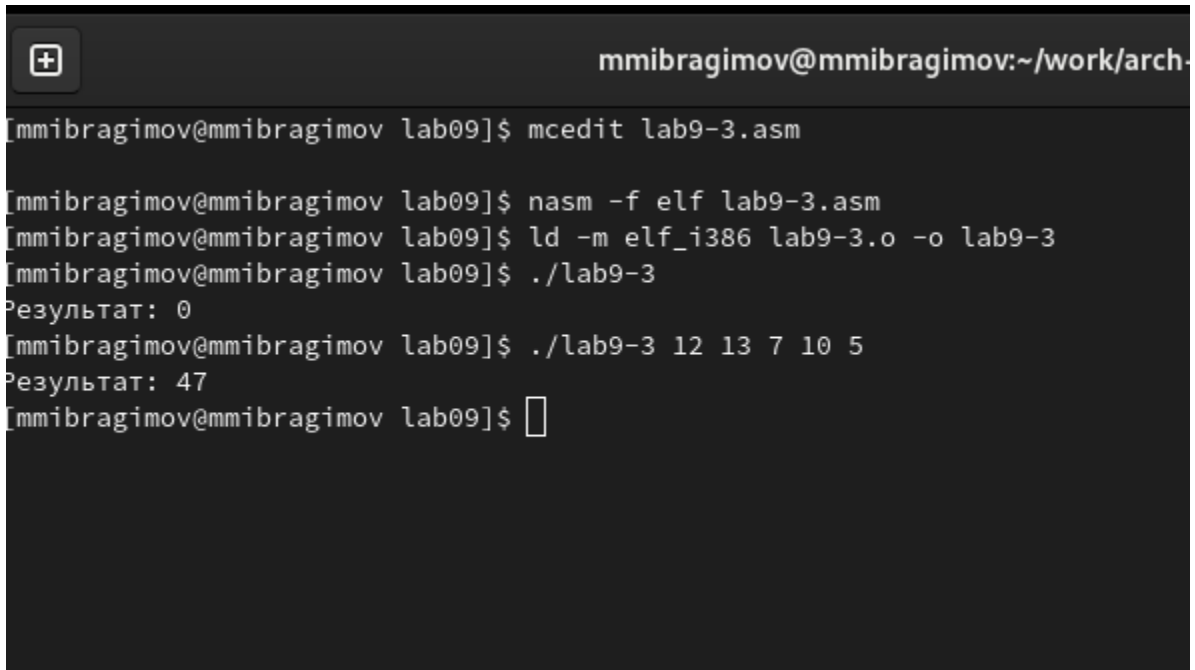
A terminal window with a dark background. The title bar shows a window icon and the text 'mmibragimov@mmibragimov:~/work/arch-'. The terminal content shows a series of commands and their outputs. The commands are: 'mcedit lab9-3.asm', 'nasm -f elf lab9-3.asm', 'ld -m elf_i386 lab9-3.o -o lab9-3', and './lab9-3'. The outputs are: 'Результат: 0' and 'Результат: 47'. The prompt is '[mmibragimov@mmibragimov lab09]\$'.

рис. 11. Результат работы программы lab9-3

Как видим, все работает корректно.

Изменим строку

add esi,ecx

на

move ebx, eax

mov eax, esi

mul ecx

mov esi, eax

а также присвоим esi значение 1, чтобы программа выводила произведение аргументов командной строки и запустим ее

Вывод:

Во время выполнения лабораторной работы были приобретены навыки написания программ с использованием циклов и обработкой аргументов командной строки.