

Шаблон отчёта по лабораторной работе

8

Бембо Жозе Лумингу

Содержание

• Цель работы	5
• Задание	6
• Теоретическое введение	7
• Выполнение лабораторной работы	9
4.1 Реализация циклов в NASM.	9
• Обработка аргументов командной строки.	14
• Задание для самостоятельной работы.	19
• Выводы	21
Список литературы	22

Список иллюстраций

4.1	создание файлов	9
•	ввод текста	10
•	запуск исполняемого файла	11
•	изменение текста программы	12
•	запуск обновленной файла.....	12
•	изменение текста программы	13
•	запуск исполняемого файла	14
•	ввод текста	15
•	запуск исполняемого файла	15
•	ввод текста	16
•	запуск исполняемого файла	17
•	изменение текста программы	18
•	запуск исполняемого файла	18
•	текст программы.....	19
•	запуск исполняемого файла	20

Список таблиц

• **Цель работы**

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

. Задание

- Реализация циклов в NASM.
- Обработка аргументов командной строки.
- Задание для самостоятельной работы.

. Теоретическое введение

- Стек — это структура данных, организованная по принципу LIFO («Last In — First Out» или «последним пришёл — первым ушёл»). Стек является частью архитектуры процессора и реализован на аппаратном уровне. Для работы со стеком в процессоре есть специальные регистры (ss, bp, sp) и команды. Основной функцией стека является функция сохранения адресов возврата и передачи аргументов при вызове процедур. Кроме того, в нём выделяется память для локальных переменных и могут временно храниться значения регистров. Стек имеет вершину, адрес последнего добавленного элемента, который хранится в регистре esp (указатель стека). Противоположный конец стека называется дном. Значение, помещённое в стек последним, извлекается первым. При помещении значения в стек указатель стека уменьшается, а при извлечении — увеличивается.
- Команда push размещает значение в стеке, т.е. помещает значение в ячейку памяти, на которую указывает регистр esp, после этого значение регистра esp увеличивается на 4. Данная команда имеет один операнд — значение, которое необходимо поместить в стек.
- Команда pop извлекает значение из стека, т.е. извлекает значение из ячейки памяти, на которую указывает регистр esp, после этого уменьшает значение регистра esp на 4. У этой команды также один операнд, который может быть регистром или переменной в памяти. Нужно помнить, что извлечённый из стека элемент не стирается из памяти и остаётся как “мусор”, который будет перезаписан при записи нового значения в стек.

- Для организации циклов существуют специальные инструкции. Для всех инструкций максимальное количество проходов задаётся в регистре есх. Наиболее простой является инструкция loop. Она позволяет организовать безусловный цикл.

. Выполнение лабораторной работы

4.1 Реализация циклов в NASM.

- 1 Создаю каталог для программ лабораторной работы № 8, перехожу в него и создаю файл lab8-1.asm.(рис.[4.1]).

```
zlbembo@fedora:~:[0]$ mkdir ~/work/arch-pc/lab08  
zlbembo@fedora:~:[0]$ cd ~/work/arch-pc/lab08  
zlbembo@fedora:~/work/arch-pc/lab08:[0]$ touch lab8-1.asm  
zlbembo@fedora:~/work/arch-pc/lab08:[0]$
```

Рис. 4.1: создание файлов

- 2 Ввожу в файл lab8-1.asm текст программы из листинга 8.1. (рис.[4.2]).

```

GNU nano 7.2                                lab8-1.asm
#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:
mov [N],ecx
mov eax,[N]
call iprintLF
loop label

call quit

```

Рис. 4.2: ввод текста

3

Создаю исполняемый файл и проверяю его работу. (рис.[4.3]).

```
zlbembo@fedora:~/work/arch-pc/lab08:[0]$ nasm -f elf lab8-1.asm
zlbembo@fedora:~/work/arch-pc/lab08:[0]$ ld -m elf_i386 -o lab8-1 lab8-1.o
zlbembo@fedora:~/work/arch-pc/lab08:[0]$ ./lab8-1
Введите N: 6
6
5
4
3
2
1
zlbembo@fedora:~/work/arch-pc/lab08:[0]$
```

Рис. 4.3: запуск исполняемого файла

4 Изменяю текст программы, добавив изменение значения регистра `ecx` в цикле. (рис.[4.4]).

```

GNU nano 7.2                                lab8-1.asm
#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:
sub ecx, 1
mov [N],ecx
mov eax,[N]
call iprintLF
loop label

call quit

```

[Lecture de 29 lignes]

Рис. 4.4: изменение текста программы

5

Создаю исполняемый файл и проверяю его работу. (рис.[4.5]).

```

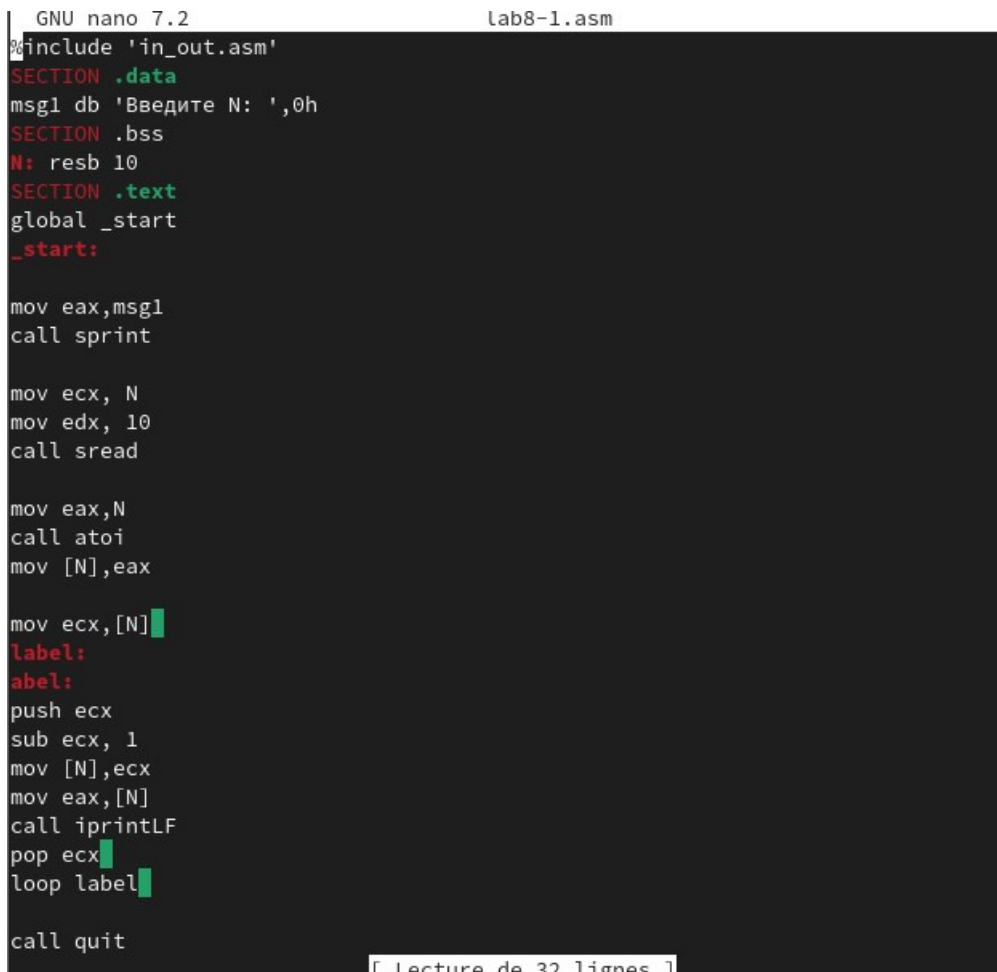
zlbembo@fedora:~/work/arch-pc/lab08:[0]$ nasm -f elf lab8-1.asm
zlbembo@fedora:~/work/arch-pc/lab08:[0]$ ld -m elf_i386 -o lab8-1 lab8-1.o
zlbembo@fedora:~/work/arch-pc/lab08:[0]$ ./lab8-1
Введите N: 10
9
7
5
3
1

```

Рис. 4.5: запуск обновленной файла

6

Вношу изменения в текст программы, добавив команды push и pop для сохранения значения счетчика цикла loop. (рис. [4.6]).



```
GNU nano 7.2                               Lab8-1.asm
%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:
abel:
push ecx
sub ecx, 1
mov [N],ecx
mov eax,[N]
call iprintLF
pop ecx
loop label

call quit
```

[Lecture de 32 lignes]

Рис. 4.6: изменение текста программы

7

Создаю исполняемый файл и проверяю его работу. (рис.[4.7]).

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

Рис. 4.7: запуск исполняемого файла

4.2 Обработка аргументов командной строки.

- 4 На этом шаге мы создали файл lab8-2.asm, затем заполнили в нем наш код.
(рис.[4.8]).

```
GNU nano 7.2      lab8-2.asm      Mod-
#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
```

Рис. 4.8: ввод текста

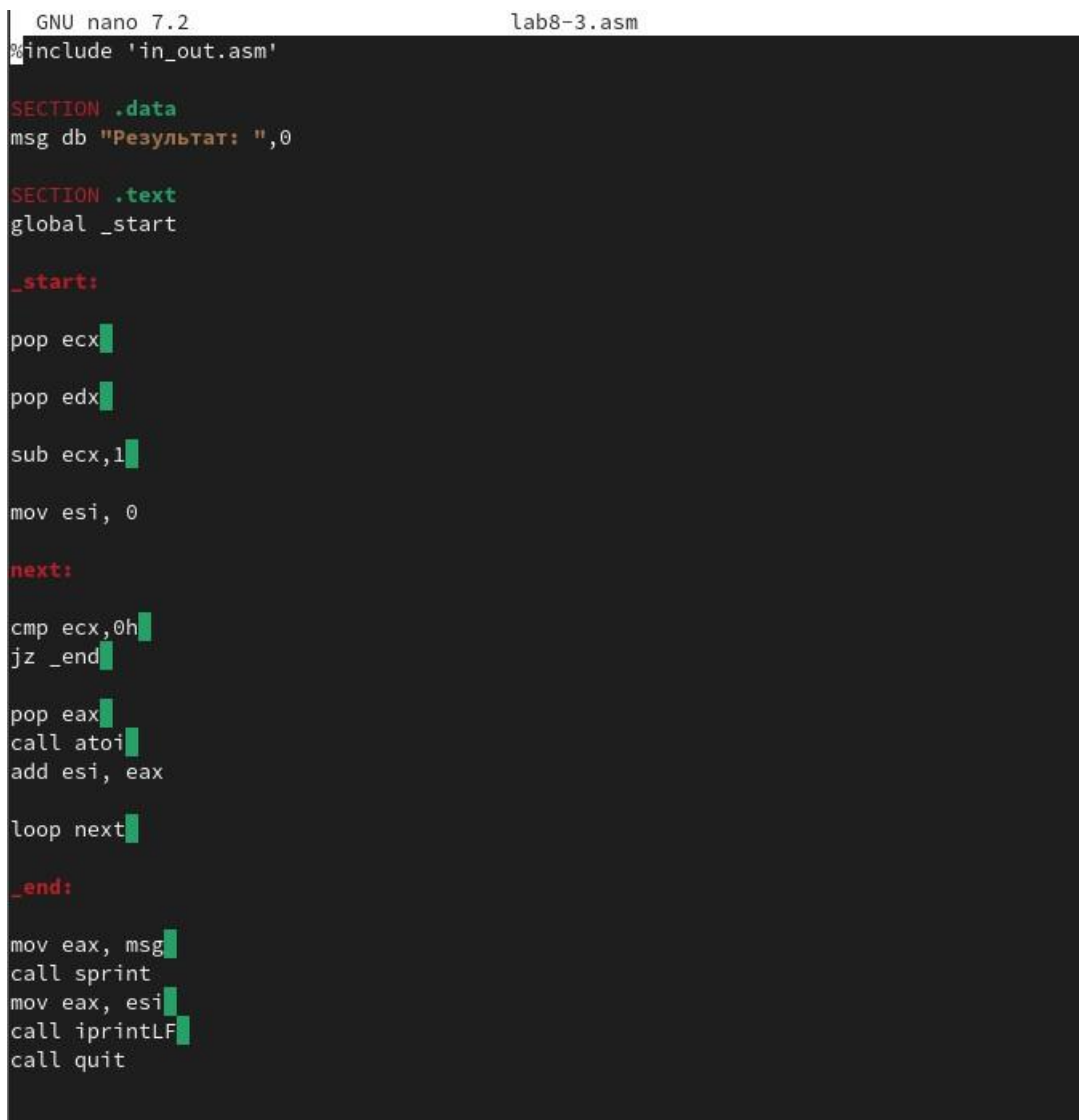
5 Создаю исполняемый файл и запускаю его, указав нужные аргументы.(рис.[4.9]).

```
zlbembo@fedora:~/work/arch-pc/lab08:[0]$ nasm -f elf lab8-2.asm
zlbembo@fedora:~/work/arch-pc/lab08:[0]$ ld -m elf_i386 -o lab8-2 lab8-2.o
zlbembo@fedora:~/work/arch-pc/lab08:[0]$ ./lab8-2 аргумент1 аргумент 2 'аргумент 3'
аргумент1
аргумент
2
аргумент 3
```

Рис. 4.9: запуск исполняемого файла

6 И, как вы можете видеть, на этот раз при запуске программы мы добавилив команду три аргумента, и в этом случае были обработаны три аргумента

7 Первым делом мы создали файл lab8-3.asm, затем заполнили кодом программы. (рис.[4.10]).



```
GNU nano 7.2 lab8-3.asm
%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
```

Рис. 4.10: ввод текста

8 Создаю исполняемый файл и запускаю его, указав аргументы.(рис.[4.11]).


```
zlbembo@fedora:~/work/arch-pc/lab08:[0]$ nasm -f elf lab8-3.asm
zlbembo@fedora:~/work/arch-pc/lab08:[0]$ ld -m elf_i386 -o lab8-3 lab8-3.o
zlbembo@fedora:~/work/arch-pc/lab08:[0]$ ./lab8-3 12 13 7 10 5
Результат: 47
```

Рис. 4.11: запуск исполняемого файла

9 Изменяю текст программы из листинга 8.3 для вычисления произведения аргументов командной строки. (рис.[4.12]).

```
GNU nano 7.2                                lab8-3.asm
%include 'in_out.asm'

SECTION .data
msg db "Результат: ",0

SECTION .text
global _start

_start:

pop ecx
pop edx
sub ecx,1
mov esi, 1

next:

cmp ecx,0h
jz _end

pop eax
call atoi

mov ebx, eax
mov eax, esi
mul ebx
mov esi, eax

loop next

_end:

mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit
```

Рис. 4.12: изменение текста программы

10Создаю исполняемый файл и запускаю его, указав аргументы. (рис.[4.13]).

```
zlbembo@fedora:~/work/arch-pc/lab08:[0]$ nasm -f elf lab8-3.asm
zlbembo@fedora:~/work/arch-pc/lab08:[0]$ ld -m elf_i386 -o lab8-3 lab8-3.o
zlbembo@fedora:~/work/arch-pc/lab08:[0]$ ./lab8-3 1 2 3 4 5
Результат: 120
```

Рис. 4.13: запуск исполняемого файла

4.3 Задание для самостоятельной работы.

- 1 В этой части мы должны были написать программу, которая находит сумму значений функции $f(x)$ для $x = x_1, x_2, \dots, x_n$
- 2 сначала мы создали наш файл lab8-4.asm, где будет находиться наш код, затем мы написали программу. (рис.[4.14]).



```
GNU nano 7.2                                lab8-4.asm
%include 'in_out.asm'

SECTION .data
msg db "Результат : ",0
msg1 db " Функция : f(x) = 30x-11",0

SECTION .text
global _start

_start:
pop ecx
pop edx
sub ecx,1
mov esi, 0

next:
cmp ecx,0h
jz _end
mov ebx,30

pop eax
call atoi

mul ebx
add eax,-11

add esi,eax
loop next

_end:
mov eax,msg1
call sprintf
mov eax,msg
call sprintf
mov eax,esi
call iprintf
call quit
```

Рис. 4.14: текст программы

- 3 Создаю исполняемый файл и проверьте его работу на нескольких наборах x

= x_1, x_2, \dots, x_n . (рис.[4.15]).

```
zlbembo@fedora:~/work/arch-pc/lab08:[0]$ nasm -f elf lab8-4.asm
zlbembo@fedora:~/work/arch-pc/lab08:[0]$ ld -m elf_i386 -o lab8-4 lab8-4.o
zlbembo@fedora:~/work/arch-pc/lab08:[0]$ ./lab8-4 1 2 3 4
Функция : f(x) = 30x-11
Результат : 256
```

Рис. 4.15: запуск исполняемого файла

. Выводы

- Благодаря этой лабораторной работе мы научились писать программы с использованием циклов и обработки аргументов командной строки, что поможет нам в дальнейшей лабораторной работе.

Список литературы

::: {#refs} :::