

Отчет по лабораторной работе №8

Дисциплина: архитектура компьютера

Хрисанова Ксения Олеговна

Содержание

1	Цель работы	1
2	Задание.....	1
3	Теоретическое введение	1
4	Выполнение лабораторной работы.....	2
4.1	Реализация циклов в NASM	2
4.2	Обработка аргументов командной строки	5
4.3	Задание для самостоятельной работы	7
5	Выводы	10
6	Список литературы	10

1 Цель работы

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

2 Задание

1. Реализация циклом в NASM
2. Обработка аргументов командной строки
3. Выполнение заданий для самостоятельной работы

3 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.

4 Выполнение лабораторной работы

4.1 Реализация циклов в NASM

Создаю каталог для программ лабораторной работы №8 (рис. 1).

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

Рис. 1: Создание каталога

Копирую в созданный файл программу из листинга. (рис. 2).



```
GNU nano 8.3 /home/kseniahrisanova/work/arch-pc/lab08/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. 2: Копирование программы из листинга

Программа показывает работу циклов в NASM (рис. 3).

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

Рис. 3: Запуск программы

Далее я изменяю значение регистра ecx (рис. 4).

```
kseniahrisnova@fedora:~/work/arch-pc/lab08 — /usr/bin/mc -P /tmp/mc.pwd.8DsVkQ
~/work/arch-pc/lab08
GNU nano 8.3 /home/kseniahrisnova/work/arch-pc/lab08/lab8-1.asm Изменён
#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
```

Рис. 4: Изменение программы

Регистр ecx на каждой итерации уменьшается на 2 значения, поэтому количество итераций уменьшается вдвое (рис. 5).

```

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

```

Рис. 5: Запуск измененной программы

Добавляю команды push и pop в программу (рис. 6).

```

GNU nano 8.3 /home/kseniahrisnova/work/arch-pc/lab08/l
#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

```

[^]G Справка [^]O Записать [^]F Поиск [^]K Вырезать [^]T Выполнить [^]C Позиция
[^]X Выход [^]R ЧитФайл [^]\ Замена [^]U Вставить [^]J Выворнять [^]/ К строке

Рис. 6: Добавление push и pop в цикл программы

Теперь количество итераций совпадает введенному N (рис.7).

```

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

```

Рис. 7: Запуск измененной программы

4.2 Обработка аргументов командной строки

Создаю новый файл для программы и копирую в него код из следующего листинга (рис. 8).

```

GNU nano 8.3 /home/kseniahrisanoval/work/arch-pc/lab08/lab8-2.asm
#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 qui

```

[^]G Справка [^]O Записать [^]F Поиск [^]K Вырезать [^]T Выполнить [^]C Позиция M-U Отмена M-A Установить
[^]X Выход [^]R ЧитФайл [^]\ Замена [^]U Вставить [^]J Выводить [^]/ К строке M-E Повтор M-6 Копировать

Рис. 8: Копирование программы из листинга

Запускаю, указав аргументы. Программа обработала то же количество аргументов, что и было введено (рис. 9).

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

Рис. 9: Запуск второй программы

Создаю новый файл для программы и копирую в него код из третьего листинга (рис. 10).

```
GNU nano 8.3 /home/kseniahrisnova/work/arch-pc/lab08/lab8-3.asm Изменён
#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. 10: Копирование программы из третьего листинга

Запускаю, указав в качестве аргументов некоторые числа, которые программа складывает (рис. 11).

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

Рис. 11: Запуск третьей программы

Изменяю программу так, чтобы она умножала аргументы, а не складывала (рис. 12).

```
GNU nano 8.3 /home/kseniahrisnova/work/arch-pc/lab08/lab8-3.asm Изменён
#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
mul esi
mov esi, eax
loop next
_end:
mov eax, msg
call sprint
mov eax, esi
call iprintLF
call quit
```

Рис. 12: Изменение третьей программы

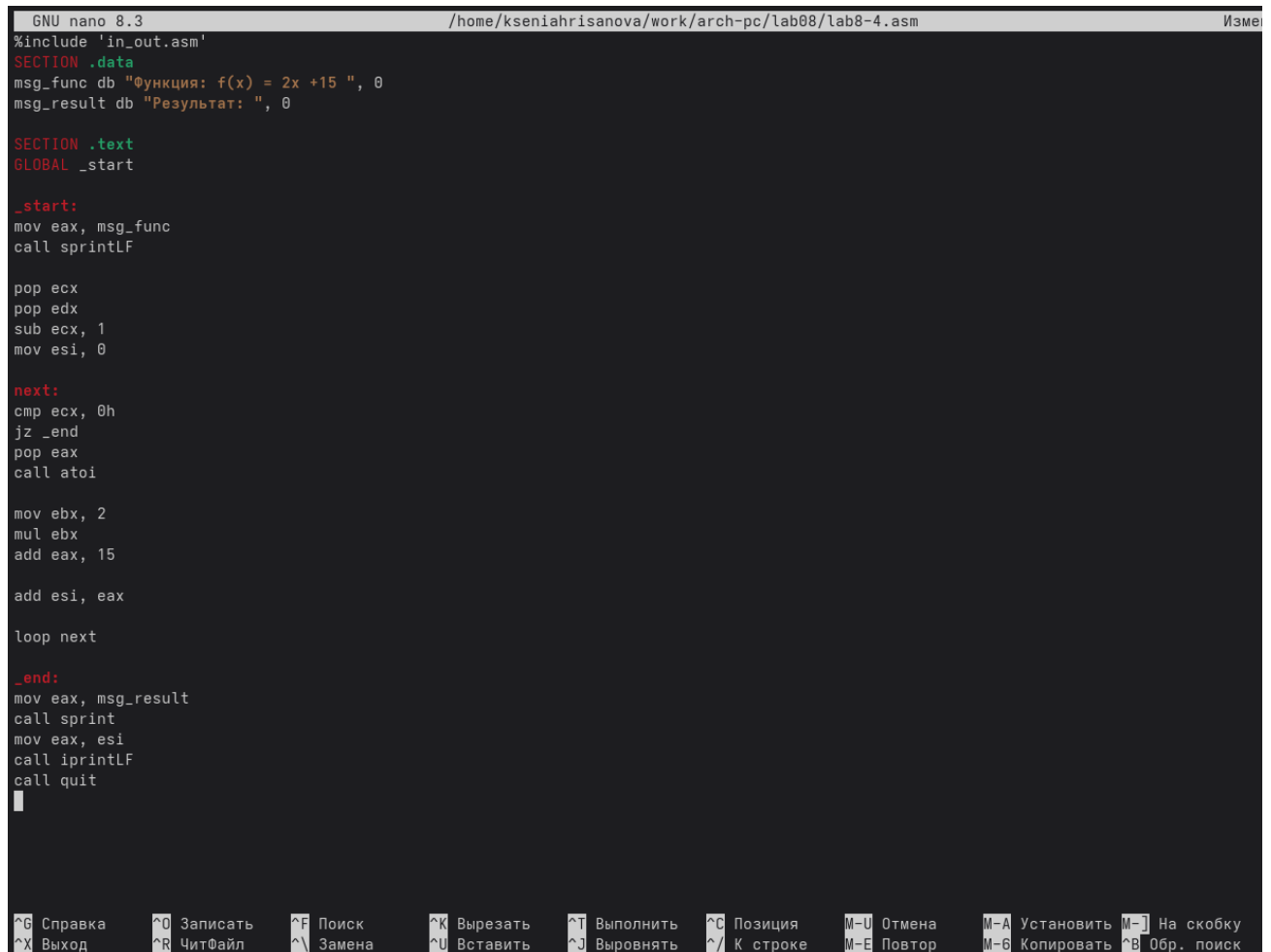
Программа теперь умножает числа (рис. 13).

```
kseniahrisnova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
kseniahrisnova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
kseniahrisnova@fedora:~/work/arch-pc/lab08$ ./lab8-3 111 1 6
Результат: 666
kseniahrisnova@fedora:~/work/arch-pc/lab08$
```

Рис. 13: Запуск измененной третьей программы

4.3 Задание для самостоятельной работы

Пишу программу, которая будет находить сумму значений для функции $f(x) = 2x + 15$ (рис. 14).



```
GNU nano 8.3 /home/kseniahrisnova/work/arch-pc/lab08/lab8-4.asm
#include 'in_out.asm'
SECTION .data
msg_func db "Функция: f(x) = 2x +15 ", 0
msg_result db "Результат: ", 0

SECTION .text
GLOBAL _start

_start:
mov eax, msg_func
call sprintLF

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

next:
cmp ecx, 0h
jz _end
pop eax
call atoi

mov ebx, 2
mul ebx
add eax, 15

add esi, eax

loop next

_end:
mov eax, msg_result
call sprint
mov eax, esi
call iprintLF
call quit
```

Рис. 14: Написание программы для самостоятельной работы

Код программы:

```
%include 'in_out.asm'

SECTION .data

msg_func db "Функция: f(x) = 2x +15 ", 0

msg_result db "Результат: ", 0

SECTION .text

GLOBAL _start

_start:

mov eax, msg_func

call sprintLF

pop ecx
```



```
pop edx
sub ecx, 1
mov esi, 0
next:
cmp ecx, 0h
jz _end
pop eax
call atoi
mov ebx, 2
mul ebx
add eax, 15
add esi, eax
loop next
_end:
mov eax, msg_result
call sprint
mov eax, esi
call iprintLF
call quit
```

Проверяю работу программы (рис. 15).

```
kseniahrisnova@fedora:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
kseniahrisnova@fedora:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
kseniahrisnova@fedora:~/work/arch-pc/lab08$ ./lab8-4 1 2 3
Функция:  $f(x) = 2x + 15$ 
Результат: 57
kseniahrisnova@fedora:~/work/arch-pc/lab08$
```

Рис. 15: Запуск программы для самостоятельной работы

5 Выводы

В результате выполнения данной лабораторной работы я приобрела навыки написания программ с использованием циклов, а также научилась обрабатывать аргументы командной строки.

6 Список литературы

1. Курс на ТУИС
2. Лабораторная работа №8