

Отчет по лабораторной работе №8

Дисциплина: Архитектура компьютера

Орлов Илья Сергеевич

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
4.1	Реализация циклов в NASM	8
4.2	Обработка аргументов командной строки	10
4.3	Задание для самостоятельной работы	12
5	Выводы	15
	Список литературы	16

Список иллюстраций

4.1	Создание каталога	8
4.2	Копирование программы из листинга	8
4.3	Запуск программы	8
4.4	Изменение программы	9
4.5	Запуск измененной программы	9
4.6	Добавление push и pop в цикл программы	9
4.7	Запуск измененной программы	10
4.8	Копирование программы из листинга	10
4.9	Запуск второй программы	10
4.10	Копирование программы из третьего листинга	11
4.11	Запуск третьей программы	11
4.12	Изменение третьей программы	11
4.13	Запуск измененной третьей программы	11
4.14	Написание программы для самостоятельной работы	12
4.15	Запуск программы для самостоятельной работы	14

Список таблиц

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. Самостоятельное написание программы по материалам лабораторной 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. -fig. 4.1).

```
rutnixya@rutnix-VirtualBox:~$ mkdir ~/work/arch-pc/lab08
rutnixya@rutnix-VirtualBox:~$ cd ~/work/arch-pc/lab08
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab08$ touch lab08-1.asm
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 4.1: Создание каталога

Копирую в созданный файл программу из листинга. (рис. -fig. 4.2).

```
GNU nano 7.2 /home/rutnixya/work/arch-pc/lab08/lab08-1.asm
#include "in_out.asm"
SECTION .data
nsqi db "Введите N: ",0h
SECTION .bss
N: resb 10
SECTION .text
global _start
_start:
mov eax,nsqi
call sprintf
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 iprintf
loop label
call quit
```

Рис. 4.2: Копирование программы из листинга

Запускаю программу, она показывает работу циклов в NASM (рис. -fig. 4.3).

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

Рис. 4.3: Запуск программы

Заменяю программу изначальную так, что в теле цикла я изменяю значение регистра есх (рис. -fig. 4.4).

```
GNU nano 7.2 /home/rutnixya/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:
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintf
loop label
call quit
```

Рис. 4.4: Изменение программы

Из-за того, что теперь регистр есх на каждой итерации уменьшается на 2 значения, количество итераций уменьшается вдвое (рис. -fig. 4.5).

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

Рис. 4.5: Запуск измененной программы

Добавляю команды push и pop в программу (рис. -fig. 4.6).

```
GNU nano 7.2 /home/rutnixya/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:
push ecx
sub ecx,1
mov [N],ecx
mov eax,[N]
call iprintf
pop ecx
loop label
call quit
```

Рис. 4.6: Добавление push и pop в цикл программы

Теперь количество итераций совпадает введенному N, но произошло смещение выводимых чисел на -1 (рис. -fig. 4.7).

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

Рис. 4.7: Запуск измененной программы

4.2 Обработка аргументов командной строки

Создаю новый файл для программы и копирую в него код из следующего листинга (рис. -fig. 4.8).

```
GNU nano 2.2 /home/rutnixya/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 sprintf
    loop next
_end:
    call quit
```

Рис. 4.8: Копирование программы из листинга

Компилирую программу и запускаю, указав аргументы. Программой было обработано то же количество аргументов, что и было введено (рис. -fig. 4.9).

```
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-2.asm
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-2 lab8-2.o
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab08$ ./lab8-2 arg1 arg 2 'arg 3'
arg1
arg
2
arg 3
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 4.9: Запуск второй программы

Создаю новый файл для программы и копирую в него код из третьего листинга (рис. -fig. 4.10).

```

GNU nano 7.2 /home/rutnixya/work/arch-pc/lab08/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,edx
loop next
_end:
mov eax,msg
call sprint
mov eax,esi
call lprintf
call quit

```

Рис. 4.10: Копирование программы из третьего листинга

Компилирую программу и запускаю, указав в качестве аргументов некоторые числа, программа их складывает (рис. -fig. 4.11).

```

rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab08$ touch lab8-3.asm
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab08$ nc

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

```

Рис. 4.11: Запуск третьей программы

Изменяю поведение программы так, чтобы указанные аргументы она умножала, а не складывала (рис. -fig. 4.12).

```

GNU nano 7.2 /home/rutnixya/work/arch-pc/lab08/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
mul esi
mov esi,edx
loop next
_end:
mov eax,msg
call sprint
mov eax,esi
call lprintf
call quit

```

Рис. 4.12: Изменение третьей программы

Программа действительно теперь умножает данные на вход числа (рис. -fig. 4.13).

```

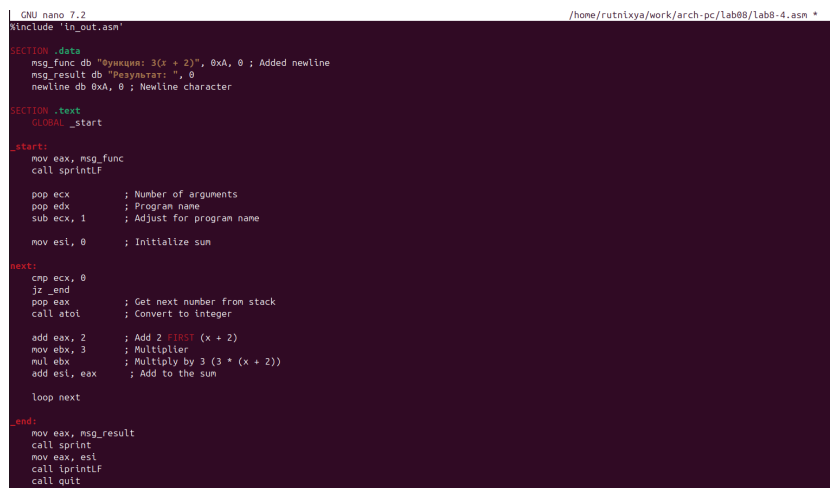
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-3.asm
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-3 lab8-3.o
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab08$ ./lab8-3 10 5 2
Результат: 100
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab08$

```

Рис. 4.13: Запуск измененной третьей программы

4.3 Задание для самостоятельной работы

Пишу программу, которая будет находить сумма значений для функции $f(x) = 3(x + 2)$, которая совпадает с моим седьмым вариантом (рис. -fig. 4.14).



```
GNU nano 7.2 /home/rutnixya/work/arch-pc/lab08/lab8-4.asm *
#include 'in_out.asm'

SECTION .data
msg_func db "Функция: 3(x + 2)", 0xA, 0 ; Added newline
msg_result db "Результат: ", 0
newline db 0xA, 0 ; Newline character

SECTION .text
GLOBAL _start

_start:
mov eax, msg_func
call sprintf

pop ecx ; Number of arguments
pop edx ; Program name
sub ecx, 1 ; Adjust for program name
mov esi, 0 ; Initialize sum

next:
cmp ecx, 0
jz _end
pop eax ; Get next number from stack
call atoi ; Convert to integer

add eax, 2 ; Add 2 FIRST (x + 2)
mov ebx, 3 ; Multiplier
mul ebx ; Multiply by 3 (3 * (x + 2))
add esi, eax ; Add to the sum

loop next

_end:
mov eax, msg_result
call sprintf
mov eax, esi
call iprintf
call quit
```

Рис. 4.14: Написание программы для самостоятельной работы

Код программы:

```
%include 'in_out.asm'
```

```
SECTION .data
```

```
msg_func db "Функция: 3(x + 2)", 0xA, 0 ; Added newline
```

```
msg_result db "Результат: ", 0
```

```
newline db 0xA, 0 ; Newline character
```

```
SECTION .text
```

```
GLOBAL _start
```

```
_start:
```

```
mov eax, msg_func
```

```
call sprintf
```

```

    pop ecx          ; Number of arguments
    pop edx          ; Program name
    sub ecx, 1       ; Adjust for program name

    mov esi, 0       ; Initialize sum

next:
    cmp ecx, 0
    jz _end
    pop eax          ; Get next number from stack
    call atoi        ; Convert to integer

    add eax, 2       ; Add 2 FIRST ( $x + 2$ )
    mov ebx, 3       ; Multiplier
    mul ebx          ; Multiply by 3 ( $3 * (x + 2)$ )
    add esi, eax     ; Add to the sum

    loop next

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

```

Проверяю работу программы, указав в качестве аргумента несколько чисел (рис. -fig. 4.15).

```
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab08$ nasm -f elf lab8-4.asm
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab08$ ld -m elf_i386 -o lab8-4 lab8-4.o
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab08$ ./lab8-4 1 2 3 4
Функция: 3(x + 2)

Результат: 54
rutnixya@rutnix-VirtualBox:~/work/arch-pc/lab08$
```

Рис. 4.15: Запуск программы для самостоятельной работы

5 Выводы

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

Список литературы

1. Курс на ТУИС
2. Лабораторная работа №8
3. Программирование на языке ассемблера NASM Столяров А. В.