

Отчет по лабораторной работе №8

Архитектура компьютера

Никифоров Захар Сергеевич

Содержание

1 Цель работы	5
2 Порядок выполнения работы	6
2.1 Реализация циклов в NASM	6
2.2 Обработка аргументов командной строки	8
3 Задание для самостоятельной работы	10
4 Выводы	13

Список иллюстраций

2.1	Скриншот терминала тс 1	6
2.2	Скриншот терминала тс 2	7
2.3	Скриншот терминала тс 3	7
2.4	Скриншот терминала тс 4	8
2.5	Скриншот терминала тс 5	8
2.6	Скриншот терминала тс 6	8
2.7	Скриншот терминала тс 7	9
2.8	Скриншот терминала тс 8	9
3.1	Скриншот терминала тс 9	10
3.2	Скриншот терминала тс 10	11
3.3	Скриншот терминала тс 11	12
3.4	Скриншот терминала тс 12	12
3.5	Скриншот терминала тс 13	12

Список таблиц

1 Цель работы

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

2 Порядок выполнения работы

2.1 Реализация циклов в NASM

Создаем каталог *lab08*, а в нем создаем файл *lab8-1.asm*, записываем код из *Листинг 8.1* и компилируем его.

Левая панель	Файл	Команда	Настройки	Пр
<- ~/work/arch-pc/lab08				.[^]>
.и	Имя		Размер	Дата правки
/..		-ВВЕРХ-		ноя 30 17:57
in_out.asm		3942	ноя 8 13:51	
*lab8-1		5024	ноя 30 18:07	
lab8-1.asm		461	ноя 30 18:06	
lab8-1.o		1440	ноя 30 18:06	

Рисунок 2.1: Скриншот терминала mc 1

```
zachnickif@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
10
9
8
7
6
5
4
3
2
1
```

Рисунок 2.2: Скриншот терминала mc 2

Получаем вывод итераций цикла от 10 до 1. Теперь внесем изменения значения регистра *esx* в цикле.

```
zachnickif@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
7
5
3
1
```

Рисунок 2.3: Скриншот терминала mc 3

Мы видим, что количество итераций стало некорректным. Оно сократилось вдвое. Снова внесем изменения, добавив команды *push* и *pop* (добавления в стек и извлечения из стека) для сохранения значения счетчика цикла *loop*.

```
zachnickif@fedora:~/work/arch-pc/lab08$ ./lab8-1
Введите N: 10
9
8
7
6
5
4
3
2
1
0
```

Рисунок 2.4: Скриншот терминала тс 4

Теперь программа работает более корректно и количество итераций равно числу, введенному с клавиатуры.

2.2 Обработка аргументов командной строки

Создадим файл *lab8-2.asm*, запишем в него текст из *Листинг 8.2*, скомпилируем и запустим, вписав перед этим три аргумента: аргумент1, аргумент2, „аргумент3“

```
zachnickif@fedora:~/work/arch-pc/lab08$ ./lab8-2 аргумент1 аргумент2 'аргумент3'
аргумент1
аргумент2
аргумент3
```

Рисунок 2.5: Скриншот терминала тс 5

Программа обработала и вывела все три аргумента. Теперь создадим новый файл *lab8-3.asm*, запишем в него текст из *Листинг 8.3*, скомпилируем и запустим с аргументами 12, 13, 7, 10, 5.

```
zachnickif@fedora:~/work/arch-pc/lab08$ ./lab8-3 12 13 7 10 5
Результат: 47
```

Рисунок 2.6: Скриншот терминала тс 6

Программа выводит результат от сложения этих чисел. Всё работает корректно. Теперь на основе этого перепишем код, чтобы программа выводила произведение.

```
lab8-3.asm      [----] 11 L:[ 1+13 14/ 24] *(208
%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
    imul esi, eax
    loop next
_end:
    mov eax, msg
    call sprint
    mov eax, esi
    call iprintLF
    call quit
....
```

Рисунок 2.7: Скриншот терминала mc 7

Я поменял значение *esi* на 1, так как, если будет 0, то результат так и будет 0. Далее нужно заменить *add* на *imul*, чтобы мы получали произведение, а не сумму.

```
zachnickif@fedora:~/work/arch-pc/lab08$ ./lab8-3 2 3 4
Результат: 24
```

Рисунок 2.8: Скриншот терминала mc 8

Получаем корректный результат.

3 Задание для самостоятельной работы

1. Создаем файл *self.asm*. Используя в качестве примера код из листинга 8.3, пишем логику, чтобы получить необходимый результат.

*self	5076	ноя 30 19:24
self.asm	589	ноя 30 19:24
self.o	1472	ноя 30 19:24

Рисунок 3.1: Скриншот терминала mc 9

```
/home/vernel/work/arch-pc/lab08/self.asm
%include 'in_out.asm'
SECTION .data
    msg1 db 'Функция: f(x) = 2x + 15', 10, 0
    msg2 db 'Результат: ', 0
SECTION .bss
    buf: RESB 80
SECTION .text
global _start
_start:
    pop ecx
    pop edx
    sub ecx, 1
    mov esi, 0
next:
    cmp ecx, 0
    jz _end

    pop ebx
    mov eax, ebx
    call atoi

    shl eax, 1
    add eax, 15

    add esi, eax

    loop next
_end:
    mov eax, msg1
    call sprint

    mov eax, msg2
    call sprint

    mov eax, esi
    call iprintLF

    mov ecx, buf
    mov edx, 80
    call sread
    call quit
```

Рисунок 3.2: Скриншот терминала mc 10

Теперь проверим корректность работы с помощью трёх троек – (2,3,4), (3,4,5), (4,5,6)

```
zachnickif@fedora:~/work/arch-pc/lab08$ ./self 2 3 4  
Функция: f(x) = 2x + 15  
Результат: 63
```

Рисунок 3.3: Скриншот терминала mc 11

```
zachnickif@fedora:~/work/arch-pc/lab08$ ./self 3 4 5  
Функция: f(x) = 2x + 15  
Результат: 69
```

Рисунок 3.4: Скриншот терминала mc 12

```
zachnickif@fedora:~/work/arch-pc/lab08$ ./self 4 5 6  
Функция: f(x) = 2x + 15  
Результат: 75
```

Рисунок 3.5: Скриншот терминала mc 13

Проверив аналитически, убеждаемся, что вывод корректный, а значит задание выполнено.

4 Выводы

В ходе работы были приобретены навыки написания программ с использованием циклов и обработкой аргументов командной строки.