

Отчёт по лабораторной работе №7

Компьютерные науки и технология программирования

Сячинова Ксения Ивановна

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Задание для самостоятельной работы	14
4	Выводы	16

Список иллюстраций

2.1	Создание каталога и файла	6
2.2	Открытие файла	6
2.3	Текст программы	7
2.4	Запуск программы	7
2.5	Изменение программы	8
2.6	Запуск программы	8
2.7	Создание файла	9
2.8	Текст программы	9
2.9	Текст программы	10
2.10	Запуск	10
2.11	Изменение	10
2.12	Создание файла	10
2.13	Текст программы	11
2.14	Запуск	11
2.15	Создание файла	12
2.16	Текст программы	12
2.17	Запуск	12
3.1	Создание файла	14
3.2	Текст программы	14
3.3	Текст программы	15
3.4	Проверка	15

Список таблиц

1 Цель работы

Освоить арифметические инструкции языка ассемблера NASM.

2 Выполнение лабораторной работы

1. Создаём каталог для программ данной лабораторной работы, создаём необходимый файл.(рис. 2.1)

```
kisyachinova1@dk3n38 ~ $ mkdir ~/work/arch-pc/lab07  
kisyachinova1@dk3n38 ~ $ cd ~/work/arch-pc/lab07  
kisyachinova1@dk3n38 ~/work/arch-pc/lab07 $ touch lab7-1.asm
```

Рис. 2.1: Создание каталога и файла

2. Затем открываем файл и вводим необходимый текст программы.(рис. 2.2),
(рис. 2.3)

```
kisyachinova1@dk3n38 ~/work/arch-pc/lab07 $ mcedit lab7-1.asm
```

Рис. 2.2: Открытие файла

```

lab7-1.asm [----]
#include 'in_out.asm'

SECTION .bss
buf1<--> RESB 80

SECTION .text
GLOBAL _start
_start:
    mov eax, '6'
    mov ebx, '4'
    add eax, ebx
    mov [buf1], eax
    mov eax, buf1
    call sprintf
    call quit

```

Рис. 2.3: Текст программы

Затем производим компиляцию файла и запускаем его. Не забываем поместить файл 'in_out.asm' в этот каталог. (рис. 2.4)

```

kisyachinova1@dk3n38 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
kisyachinova1@dk3n38 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
kisyachinova1@dk3n38 ~/work/arch-pc/lab07 $ ./lab7-1
j
kisyachinova1@dk3n38 ~/work/arch-pc/lab07 $

```

Рис. 2.4: Запуск программы

Вместо 10 в данном случае у нас выводится 'j'. Дело в том, что код символа '6' равен 00110110 в двоичном представлении, код символа '4' равен 00110100. Команда складывает эти коды и получается 01101010, что в свою очередь является кодом символа 'j'.

3. Изменим пару строк в программе. (рис. 2.5), (рис. 2.6)

```

lab7-1.asm [-
#include 'in_out.asm'

SECTION .bss
buf1:<->RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax, 6
mov ebx, 4
add eax, ebx
mov [buf1], eax
mov eax, buf1
call sprint

call quit

```

Рис. 2.5: Изменение программы

```

kisyachinova1@dk6n58 ~/work/arch-pc/lab07 $ nasm -f elf lab7-1.asm
kisyachinova1@dk6n58 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-1 lab7-1.o
kisyachinova1@dk6n58 ~/work/arch-pc/lab07 $ ./lab7-1
kisyachinova1@dk6n58 ~/work/arch-pc/lab07 $ █

```

Рис. 2.6: Запуск программы

После запуска программы видим, что символ не отображается на экране. Согласно таблице ASCII имеем символ 'STX'.

4. Создаём файл 'lab7-2.asm' и открываем его.(рис. 2.7)


```
kisyachinova1@dk6n58 ~/work/arch-pc/lab07 $ touch lab7-2.asm
kisyachinova1@dk6n58 ~/work/arch-pc/lab07 $ mcedit lab7-2.asm
```

Рис. 2.7: Создание файла

Пишем листинг программы. (рис. 2.8)

```
lab7-2.asm [----] 13
#include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:

    mov eax, '6'
    mov ebx, '4'
    add eax, ebx
    call iprintLF
    ....
    call quit
```

Рис. 2.8: Текст программы

В результате мы получаем число 6, так как команда `add` складываем коды символов. Однако команда `'iprintLF'` позволяет вывести именно число, а не символ, кодом которого является это число.

5. Аналогично заменим символы на числа. Заменим определённые строки, запустим файл. (рис. 2.9), (рис. 2.10)

```

lab7-2.asm      [-M--] 13 L: 1+ 6
#include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:

    mov eax,6
    mov ebx,4
    add eax,ebx
    call iprintLF
    ....
    call quit

```

Рис. 2.9: Текст программы

```

kisyachinova1@dk6n58 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
kisyachinova1@dk6n58 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
kisyachinova1@dk6n58 ~/work/arch-pc/lab07 $ ./lab7-2
10
kisyachinova1@dk6n58 ~/work/arch-pc/lab07 $

```

Рис. 2.10: Запуск

В результате получили число 10. При изменении 'iprintLF' на 'iprint' результат не выводится на новую строку(рис. 2.11)

```

kisyachinova1@dk6n58 ~/work/arch-pc/lab07 $ nasm -f elf lab7-2.asm
kisyachinova1@dk6n58 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-2 lab7-2.o
kisyachinova1@dk6n58 ~/work/arch-pc/lab07 $ ./lab7-2
10kisyachinova1@dk6n58 ~/work/arch-pc/lab07 $

```

Рис. 2.11: Изменение

- Создаем файл 'lab7-3.asm' и вводим текст программы. Данная программа позволяет вычислять значения выражения $\Box(\Box) = (4 \Box 6 + 2)/5$.(рис. 2.12) (рис. 2.13), (рис. 2.14)

```

kisyachinova1@dk6n58 ~/work/arch-pc/lab07 $ touch lab7-3.asm
kisyachinova1@dk6n58 ~/work/arch-pc/lab07 $ mcedit lab7-3.asm

```

Рис. 2.12: Создание файла

```

lab7-3.asm      [----]  9 L:  1+3
%include 'in_out.asm'

SECTION .data
div: DB 'Результат:',0
rem: DB 'Остаток от деления: ',0

SECTION .text
GLOBAL _start
_start:

mov eax,5
mov ebx,2
mul ebx
add eax,3
xor edx,edx
mov ebx,3
div ebx

mov edi,eax

mov eax,div
call sprint
mov eax,edi
call iprintLF

mov eax,rem
call sprint
mov eax,edx
call iprintLF

call quit

```

Рис. 2.13: Текст программы

```

kisyachinova1@dk6n58 ~/work/arch-pc/lab07 $ nasm -f elf lab7-3.asm
kisyachinova1@dk6n58 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-3 lab7-3.o
kisyachinova1@dk6n58 ~/work/arch-pc/lab07 $ ./lab7-3
Результат:4
Остаток от деления: 1
kisyachinova1@dk6n58 ~/work/arch-pc/lab07 $ 

```

Рис. 2.14: Запуск

7. Рассмотрим ещё одну программу, которая позволит определить вариант с помощью номера студенческого билета. Для этого создаём файл, пишем программу и запускаем её. (рис. 2.15), (рис. 2.16), (рис. 2.17)

```
kisyachinova1@dk6n52 ~/work/arch-pc/lab07 $ touch variant.asm
kisyachinova1@dk6n52 ~/work/arch-pc/lab07 $ mcedit variant.asm
```

Рис. 2.15: Создание файла

```
variant.asm [----] 10 L: [ 1+16 17/ 34] *(247
#include 'in_out.asm'

SECTION .data
msg: DB 'Введите № студенческого билета: ',0
rem: DB 'Ваш вариант: ',0

SECTION .bss
x: <----> RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax, msg
call sprintf

mov ecx, x
mov edx, 80
call sread

mov eax, x
call atoi

xor edx, edx
mov ebx, 20
div ebx
inc edx

mov eax, rem
call sprintf
mov eax, edx
call iprintfLF

call quit
```

Рис. 2.16: Текст программы

```
kisyachinova1@dk6n52 ~/work/arch-pc/lab07 $ mcedit variant.asm
kisyachinova1@dk6n52 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o variant variant.o
kisyachinova1@dk6n52 ~/work/arch-pc/lab07 $ ./variant
Введите № студенческого билета:
1132226510
Ваш вариант: 11
kisyachinova1@dk6n52 ~/work/arch-pc/lab07 $
```

Рис. 2.17: Запуск

Результат: мой вариант №11.

- Ответы на вопросы:

1. Какие строки листинга 7.4 отвечают за вывод на экран сообщения 'Ваш вариант: '? Строки `mov eax, rem` `call sprintf`

2. Для чего используются следующие инструкции `asm "mov ecx, x"`, `"mov edx, 80"`, `"call sread"`? Инструкция `"mov ecx, x"` записывает адреса выводимого сообщения в 'EAX'. Инструкция `"mov edx, 80"` записывает длину вводимого сообщения в 'EBX'. Инструкция `"call sread"` выполняет вызов программы ввода сообщения.
3. Для чего используется инструкция `"call atoi"`? Данная конструкция используется для преобразования символа в число.
4. Какие строки листинга 7.4 отвечают за вычисления варианта? `xor edx,edx`
`mov ebx,20` `div ebx` `inc edx`
5. В какой регистр записывается остаток от деления при выполнении инструкции `"div ebx"`? Остаток от деления при выполнении данной инструкции записывается в регистр 'EBX'.
6. Для чего используется инструкция `"inc edx"`? Данная конструкция используется для увеличения значения `edx` на единицу.
7. Какие строки листинга 7.4 отвечают за вывод на экран результата вычислений? `mov eax, edx` `call iprintLF`

3 Задание для самостоятельной работы

1. Нужно написать программу для выражения $f(x) = 10(x + 1) - 10$ (вариант 11).
При значениях $x_1=1$, $x_2=7$.

Для начала создадим файл, в котором напишем код программы и откроем его.
(рис. 3.1)

```
kisyachinova1@dk8n74 ~/work/arch-pc/lab07 $ touch lab7-4.asm
kisyachinova1@dk8n74 ~/work/arch-pc/lab07 $ mcedit lab7-4.asm
```

Рис. 3.1: Создание файла

После вводим текст программы. (рис. 3.2), (рис. 3.3)

```
%include 'in_out.asm'

SECTION .data
rem: DB 'Вычислить значение выражения 10(x+1)-10',0
msg: DB 'Введите x: ',0
div: DB 'Результат: ',0

SECTION .bss
x:<---->RESB 80
rez:<-->RESB 80

SECTION .text
GLOBAL _start
_start:

mov eax, rem
call sprintf

mov eax, msg
call sprintf

mov ecx, x
mov edx, 80
call read
```

Рис. 3.2: Текст программы

```

mov eax, x
call atoi

mov ebx, 1
add eax, ebx
mov ebx, 10
mul ebx
sub eax, 10

mov [rez], eax

mov eax, div
call sprintf
mov eax, [rez]
call iprintf

call quit

```

Рис. 3.3: Текст программы

После этого проводим компиляцию файла и проверяем выполнение работы.
Всё верно. (рис. 3.4)

```

kisyachinova1@dk8n74 ~/work/arch-pc/lab07 $ nasm -f elf lab7-4.asm
kisyachinova1@dk8n74 ~/work/arch-pc/lab07 $ ld -m elf_i386 -o lab7-4 lab7-4.o
kisyachinova1@dk8n74 ~/work/arch-pc/lab07 $ ./lab7-4
Вычислить значение выражения 10(x+1)-10
Введите x:
1
Результат:
10
kisyachinova1@dk8n74 ~/work/arch-pc/lab07 $ ./lab7-4
Вычислить значение выражения 10(x+1)-10
Введите x:
7
Результат:
70

```

Рис. 3.4: Проверка

4 Выводы

В ходе выполнения данной лабораторной работы я освоила арифметические инструкции языка ассемблера NASM.