

Лабораторная работа №6

Арифметические операции в NASM

Коровкин Никита Михайлович

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выполнение самостоятельной работы	13
4	Выводы	16

Список иллюстраций

2.1	Создание файла lab6-1.nasm	6
2.2	файл в Midnight Commander	6
2.3	Записываем необходимый код в файл	6
2.4	Добавляем файл для корректной работы кода	7
2.5	Запуск файла lab6-1.nasm	7
2.6	Редактируем код файла lab6-1.nasm	7
2.7	Повторный запуск кода	8
2.8	Создание файла lab6-2.nasm	8
2.9	Записываем необходимый код в файл	8
2.10	Запуск кода	8
2.11	Запуск отредактированного кода	9
2.12	Заменяем в коде iprintLF на iprint	9
2.13	Запуск кода	9
2.14	Создание файла lab6-3.nasm	9
2.15	Вставляем необходимый код	10
2.16	Проверяем работу кода	10
2.17	Запуск отредактированного кода	10
2.18	Файл variant.asm	11
2.19	Запуск кода для вычитывания варианта	11
3.1	Создание файл для самостоятельной работы	13
3.2	Код для вычисления	14
3.3	Первый запуск кода	14
3.4	Второй запуск кода	15

Список таблиц

1 Цель работы

Познакомиться с базовыми инструкциями языка Ассемблер, отвечающими за основные арифметические операции.

2 Выполнение лабораторной работы

В первую очередь переместимся в папку для 6-й лабораторной работы и создадим первый файл.(рис.1)

```
liveuser@localhost-live:~$ cd ~/work/arch-pc/lab06
liveuser@localhost-live:~/work/arch-pc/lab06$ touch lab6-1.asm
liveuser@localhost-live:~/work/arch-pc/lab06$
```

Рис. 2.1: Создание файла lab6-1.asm

Запустим Midnight Commander и откроем наш файл.(рис.2)

.n	Name	Size	Modify time	.n	Name	Size	Modify time
/..		UP--DIR	Nov 11 02:59	/..		UP--DIR	Nov 11 02:59
lab6-1.asm		0	Nov 11 03:01	lab6-1.asm		0	Nov 11 03:01

Рис. 2.2: файл в Midnight Commander

Открыв файл, вставляем первый код из листинга 6.1.(рис.3)

```
GNU nano 7.2 /home/liveuser/work/arch-pc/lab06/lab6-1.asm Modified
%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 sprintLF
call quit
```

Рис. 2.3: Записываем необходимый код в файл

После этого добавим в нашу папку файл in_out.asm, так как без него наш код работать не будет.(рис.4)

Left	File	Command	Options	Right
<-	~/work/arch-pc/lab06			<- ~/work/arch-pc/lab06
.n	Name	Size	Modify time	.n Name Size Modify time
./..	UP--DIR	Nov 11 02:59		./.. UP--DIR Nov 11 02:59
in_out.asm	3942	Nov 7 10:08		lab6-1.asm 173 Nov 11 03:25
lab6-1.asm	173	Nov 11 03:25		

Рис. 2.4: Добавляем файл для корректной работы кода

Теперь запустим код из нашего первого файла. На выходе мы получим j. Это неправильный вывод, так как изначально мы складывали цифры.

```
liveuser@localhost-live:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
liveuser@localhost-live:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
liveuser@localhost-live:~/work/arch-pc/lab06$ ./lab6-1
j
```

Рис. 2.5: Запуск файла lab6-1.nasm

Чтобы получить на выходе верный результат, отредактируем код, убрав кавычки.(рис.6)

```
mov eax,6
mov ebx,4
add eax,ebx
mov [buf1],eax
```

Рис. 2.6: Редактируем код файла lab6-1.nasm

Теперь запустим код еще раз. НА выходе мы ничего не видим, однако это не так. По таблице ASCII под 10 номером обозначен перевод строки. Наш код выводит не число 10, а символ с данным номером.(рис.7)

```
liveuser@localhost-live:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
liveuser@localhost-live:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
liveuser@localhost-live:~/work/arch-pc/lab06$ ./lab6-1
```

Рис. 2.7: Повторный запуск кода

Создадим второй файл - lab6-2.asm.(рис.8)

```
liveuser@localhost-live:~/work/arch-pc/lab06$ touch lab6-2.asm
liveuser@localhost-live:~/work/arch-pc/lab06$
```

Рис. 2.8: Создание файла lab6-2.asm

Вставим в него код из листинга 6.2.(рис.9)

```
%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: Записываем необходимый код в файл

Запускаем код и на выходе получаем 106.(рис.10)

```
liveuser@localhost-live:~/work/arch-pc/lab06$ ./lab6-2
106
```

Рис. 2.10: Запуск кода

Мы получили 106, так как цифры были в кавычках и мы получили сложение их номеров. 54 и 52 в сумме дают 106. Тогда уберем кавычки и проверим программу.(рис.11)


```
liveuser@localhost-live:~/work/arch-pc/lab06$ ./lab6-2
10
```

Рис. 2.11: Запуск отредактированного кода

Все выводится верно. Теперь отредактируем код еще раз. Заменим в коде `iprintLF` на `iprint`, чтобы узнать отличие.(рис.12)

```
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprint
call quit
```

Рис. 2.12: Заменяем в коде `iprintLF` на `iprint`

Теперь запустим файл.(рис.13)

```
liveuser@localhost-live:~/work/arch-pc/lab06$ ./lab6-2
10liveuser@localhost-live:~/work/arch-pc/lab06$
```

Рис. 2.13: Запуск кода

Как мы видим, после замены число выводится без переноса строки.

Создадим теперь третий файл.(рис.14)

```
10liveuser@localhost-live:~/work/arch-pc/lab06$ touch lab6-3.asm
liveuser@localhost-live:~/work/arch-pc/lab06$
```

Рис. 2.14: Создание файла `lab6-3.nasm`

Вставим в него код из листинга 6.3.(рис.15)

```
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax,5 ; EAX=5
mov ebx,2 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,3 ; EAX=EAX+3
xor edx,edx ; обнуляем EDX для корректной работы div
```

Рис. 2.15: Вставляем необходимый код

Код должен выводить значение функции $(5*2+3)/3$. Запустим код. На выходе у нас ответ 4 и остаток от деления 1.(рис.16)

```
liveuser@localhost-live:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
```

Рис. 2.16: Проверяем работу кода

Наш результат совпадает с результатом, указанным в лабораторной работе. Теперь изменим файл так, чтобы он вычислял значение выражения $(4*6+2)/5$. Запустим код и проверим, совпадет ли результат.(рис.17)

```
liveuser@localhost-live:~/work/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
```

Рис. 2.17: Запуск отредактированного кода

Результат действительно совпадает.

Теперь нам необходимо создать файл, который будет высчитывать вариант задания для самостоятельной работы. Вставляем необходимый код в файл.(рис.18)

```

%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 sprintLF
mov ecx, x
mov edx, 80
call sread
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`
xor edx, edx
mov ebx, 20

```

Рис. 2.18: Файл variant.asm

Теперь запустим код. Введем туда номер студенческого билета.(рис.19)

```

liveuser@localhost-live:~/work/arch-pc/lab06$ nasm -f elf variant.asm
liveuser@localhost-live:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
liveuser@localhost-live:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132246835
Ваш вариант: 16
liveuser@localhost-live:~/work/arch-pc/lab06$

```

Рис. 2.19: Запуск кода для высчитывания варианта

Мы получили на выходе 16. Таким будет номер нашего варианта.

Разберем то, как работает код, ответив на следующие вопросы.

1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения 'Ваш вариант:'?

За вывод сообщения отвечает call sprint и строка mov eax,rem, перемещающая строку с фразой в регистр eax, из которого мы считываем данные для вывода

2. Для чего используются следующие инструкции?

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

Данные строки используются для записи данных в переменную x.

3. Для чего используется инструкция “call atoi”?

Инструкция используется для преобразования ASCII кода в число.

4. Какие строки листинга 6.4 отвечают за вычисления варианта?

```
div ebx и inc edx
```

Первая отвечает за деление, вторая - прибавляет единицу

5. В какой регистр записывается остаток от деления при выполнении инструкции “div ebx”?

В регистр edx

6. Для чего используется инструкция “inc edx”?

Оно увеличивает значение на единицу

7. Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений?

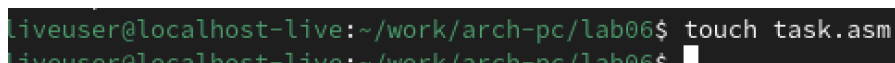
```
mov eax,edx и call iprintLF
```

Первая строка переносит значение регистра edx в eax, а вторая выводит значение регистра eax

3 Выполнение самостоятельной работы

В качестве самостоятельной работы решим задание из 16-го варианта. Это уравнение $(10x_1 - 5)^2$, где $x_1 = 1$, $x_2 = 3$.

Для начала создадим файл для вычисления.(рис.20)



```
liveuser@localhost-live:~/work/arch-pc/lab06$ touch task.asm
```

Рис. 3.1: Создание файл для самостоятельной работы

Откроем файл и запишем туда код для вычисления.(рис.21)

```
GNU nano 1.2 /home/liveuser/work/arch-pc/lab06/task.asm
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg: DB '(10x-5)^2',0
msg2: DB 'Enter x: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax, msg
call sprintLF
mov eax, msg2
call sprintLF
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov ebx, 10
[ Read 26 lines ]
^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line
```

Рис. 3.2: Код для вычисления

Первые 4 строки отвечают за вывод сообщения на экран.

Следующие 3 строки мы используем для ввода значения переменной x.

mov eax, x и call atoi отвечают за преобразование ASCII кода в число.

Затем умножаем x на 10 и вычитаем 5. Полученное число умножаем на само себя, чтобы возвести в квадрат.

В конце выводим результат и завершаем программу.

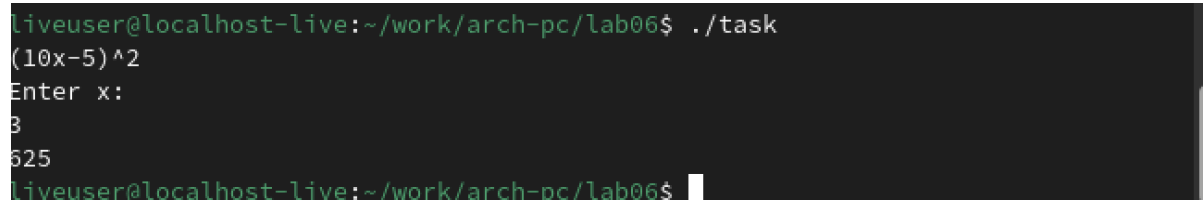
Запустим код и проверим результат.(рис.22)

```
liveuser@localhost-live:~/work/arch-pc/lab06$ nasm -f elf task.asm
liveuser@localhost-live:~/work/arch-pc/lab06$ ld -m elf_i386 -o task task.o
liveuser@localhost-live:~/work/arch-pc/lab06$ ./task
(10x-5)^2
Enter x:
1
25
```

Рис. 3.3: Первый запуск кода

Введя 1, получаем 25. Посчитав вручную, мы понимаем, что ответ верный.

Запусти код второй раз, введя 3.(рис.23)



```
liveuser@localhost-live:~/work/arch-pc/lab06$ ./task
(10x-5)^2
Enter x:
3
625
liveuser@localhost-live:~/work/arch-pc/lab06$
```

Рис. 3.4: Второй запуск кода

Ответ 625, значит код написан верно.

4 Выводы

В ходе выполнения лабораторной работы мы получили знания о видах и работе арифметических операций на языке Ассемблера. Также мы самостоятельно написали программу, закрепив знания.