

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ
Факультет физико-математических и естественных наук
Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ №6
дисциплина: Архитектура компьютера

Студент: Волчкова Елизавета Дмитриевна

Группа: НКАбд-01-24

МОСКВА

2024 г.

1.	Цель работы	3
2.	Теоретическое введение	4
3.	Задания для самостоятельной работы	5
4.	Выполнение лабораторной и самостоятельной работ	6
5.	Вывод	14
6.	Список литературы.	15
.		

Цель работы.

Узнать и ознакомиться с арифметическими инструкциями языка ассемблера NASM.

Теоретическое введение.

Адресация в NASM Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти.

Далее рассмотрены все существующие способы задания адреса хранения операндов – способы адресации.

Существует три основных способа адресации:

- Регистровая адресация – операнды хранятся в регистрах и в команде используются имена этих регистров, например: `mov ax,bx`.

- Непосредственная адресация – значение операнда задается непосредственно в команде,

Например: `mov ax,2`.

- Адресация памяти – операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию.

Задание для самостоятельной работы.

1. Написать программу вычисления выражения $y = f(x)$.
2. Программа должна выводить выражение для вычисления, выводить запрос на ввод значения x , вычислять заданное выражение в зависимости от введенного x , выводить результат вычислений.
3. Вид функции $f(x)$ выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы.
4. Создайте исполняемый файл и проверьте его работу для значений x_1 и x_2 из 6.3

Выполнение лабораторной работы.

Создала каталог для программ лабораторной работы № 6, перешла в него и создала файл lab6-1.asm:

```
mkdir ~/work/arch-pc/lab06  
cd ~/work/arch-pc/lab06  
touch lab6-1.asm
```

```
This message is shown once a day. To disable it please create the  
/root/.hushlogin file.  
root@LAPTOP-KE7VHG2Q:~# mkdir ~/work/arch-pc/lab06  
root@LAPTOP-KE7VHG2Q:~# cd ~/work/arch-pc/lab06  
root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab06# touch lab06-1.asm  
root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab06#
```

2. Рассмотрела примеры программ вывода символьных и численных значений. Программы будут выводить значения записанные в регистры `eax`.

Затем ввела в файл lab6-1.asm текст программы из листинга 6.1. В данной программе в регистр `eax` записала символ 6 (`mov eax,'6'`), в регистр `ebx` символ 4 (`mov ebx,'4'`).

Далее к значению в регистре `eax` прибавила значение регистра `ebx` (`add eax,ebx`, результат сложения запишется в регистр `eax`).

Потом вывела результат. Записала значение регистра `eax` в переменную `buf1` (`mov [buf1],eax`), а потом адрес переменной `buf1` в регистр `eax` (`mov eax,buf1`) и вызовом функции `sprintf`.

Ввела программу вывода значения регистра `eax`, создала исполняемый файл и запустила его.

```
nasm -f elf lab6-1.asm  
ld -m elf_i386 -o lab6-1 lab6-1.o  
./lab6-1
```

```

mc [root@LAPTOP-KE7VHG2Q]:~/work/arch-pc/lab06
Left  File      Command  Options  Right
<- ~/work/arch-pc .[^]> <- ~/work/arch-pc/lab06 .
.n      Name      Size      Modify   time
UP--DIR Oct 26 00:01
/lab04   4096 Nov 7 23:05
/lab05   4096 Nov 9 21:51
/lab06   4096 Nov 15 21:48

/lab06 1005G/1007G (99%)
UP--DIR 1005G/1007G (99%)

Hint: Leap to frequently used directories in a single bound with C-.
root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab06# ./lab6-1
1Help 2Menu 3View 4Edit 5Copy 6RenMov 7Mkdir 8Delete 9PullDn 10Quit

```

После изменила текст программы и вместо символов, записала в регистры числа. Исправила текст программы (Листинг 6.1) следующим образом: заменила строки

```

mov eax,'6'

mov ebx,'4' на строки

mov eax,6

mov ebx,4

```

```

GNU nano 6.2 lab6-1.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 printf
call quit

```

Создала исполняемый файл и запустила его.

Преобразовала текст программы из Листинга 6.1 с использованием этих функций. Создала файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и ввела в него текст программы из листинга 6.2.

```
touch ~/work/arch-pc/lab06/lab6-2.asm
```

Создала исполняемый файл и запустила его.

```
mc [root@LAPTOP-KE7VHG2Q]:~/work/arch-pc/lab06
GNU nano 6.2 lab
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit

^G Help      ^O Write Out  ^W Where Is   ^K Cut
^X Exit      ^R Read File  ^\ Replace    ^U Paste
```

~/work/arch-pc/lab06		.[^]>	
.n	Name	Size	Modify time
/..		UP--DIR	Nov 15 21:34
	.lab6-2.asm.swp	1024	Nov 15 22:08
	*in_out.asm	3942	Nov 7 23:15
	*lab6-1	5160	Nov 15 21:48
	*lab6-1.asm	182	Nov 15 21:58
	lab6-1.o	1200	Nov 15 21:55
	*lab6-2.asm	127	Nov 15 22:12

В результате работы программы я получила число 106

Аналогично предыдущему примеру изменила символы на числа. Потом заменила строки `mov eax,'6'` `mov ebx,'4'` на строки `mov eax,6` `mov ebx,4` Создала исполняемый файл и запустила его

Заменила функцию `iprintLF` на `iprint`, создала исполняемый файл и запустила его.


```

GNU nano 6.2
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprint
call quit

```

Создала файл lab6-3.asm в каталоге ~/work/arch-pc/lab06:

```
touch ~/work/arch-pc/lab06/lab6-3.asm
```

```

root@LAPTOP-KE7VHG2Q:~# touch ~/work/arch-pc/lab06/lab6-3.asm
root@LAPTOP-KE7VHG2Q:~#

```

Изучила текст программы из листинга 6.3 и ввела в lab6-3.asm.
Создала исполняемый файл и запустила его.

Результат работы программы должен быть следующим:
user@dk4n31:~\$./lab6-3

Результат: 4

Остаток от деления: 1

user@dk4n31:~\$

Изменила текст программы для вычисления выражения

```

GNU nano 6.2                                lab6-3.asm
#include 'in_out.asm'
section .data
    result db 'Result: %d', 10, 0 ; Формат строки для вывода

section .bss
    res resd 1 ; Резервируем место для результата

section .text
    global _start

_start:
    ; Вычисляем f(x) = (4 * 6 + 2) / 5
    mov eax, 4 ; Загружаем 4 в регистр EAX
    imul eax, 6 ; Умножаем EAX на 6 (EAX = 24)
    add eax, 2 ; Добавляем 2 (EAX = 26)
    mov ebx, 5 ; Загружаем 5 в регистр EBX
    xor edx, edx ; Обнуляем EDX перед делением
    div ebx ; Делим EAX на EBX (EAX = 26 / 5 = 5, EDX = 1)

    ; Сохраняем результат
    mov [res], eax ; Сохраняем результат в переменной res

    ; Выводим результат
    push dword [res] ; Параметр для printf
    push result ; Строка формата
    call iprintf ; Вызов функции printf
    add esp, 8 ; Очистка стека

    ; Завершение программы
    mov eax, 1 ; Системный вызов для выхода
    xor ebx, ebx ; Код возврата 0
    int 0x80 ; Вызов ядра

[ Read 32 lines (Converted from DOS format) ]
^G Help      ^O Write Out  ^W Where Is   ^K Cut        ^T Execute    ^C Location   M-U Undo
^X Exit      ^R Read File  ^\ Replace    ^U Paste      ^J Justify    ^/ Go To Line M-E Redo

```

$f(x) = (4 * 6 + 2)/5$. Создала исполняемый файл и проверила его работу. Ответ: 5!

```

root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab06# ./lab6-3
5
root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab06#

```

Далее вывела запрос на введение № студенческого билета, потом
вычислила номер варианта по формуле:

```
root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab06# nasm -f elf lab6-4.asm
root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab06# ld -m elf_i386 -o lab6-4 lab6-4.o
root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab06# ./lab6-4
Введите № студенческого билета:
1132246759
Ваш вариант: 20
root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab06#
```

$(S_n \bmod 20) + 1$, где S_n – номер студенческого билета (В данном случае $a \bmod b$ – это остаток от деления a на b), потом вывела на экран номер варианта.

Создала файл variant.asm в каталоге ~/work/arch-pc/lab06:

```
touch ~/work/arch-pc/lab06/variant.asm
```

Изучила текст программы из листинга 6.4 и ввела в файл variant.asm.

Потом создала исполняемый файл и запустила его, далее проверила результат работы программы, вычислив номер варианта аналитически.

Ответы на вопросы:

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

20

2. Для чего используются следующие инструкции?

mov esx, x помещаем в регистр esx число x

mov edx, 80 длина строки 80 символов

call sread считывает из строки длиной 80 символов значение переменной x

3. Для чего используется инструкция “call atoi”?

4. Какие строки листинга 6.4 отвечают за вычисления варианта?

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

При этом остаток от целочисленного деления помещается в регистр AH, DX или EDX соответственно.

6. Для чего используется инструкция “inc edx”? -Инструкция inc edx используется для увеличения значения регистра EDX на 1

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

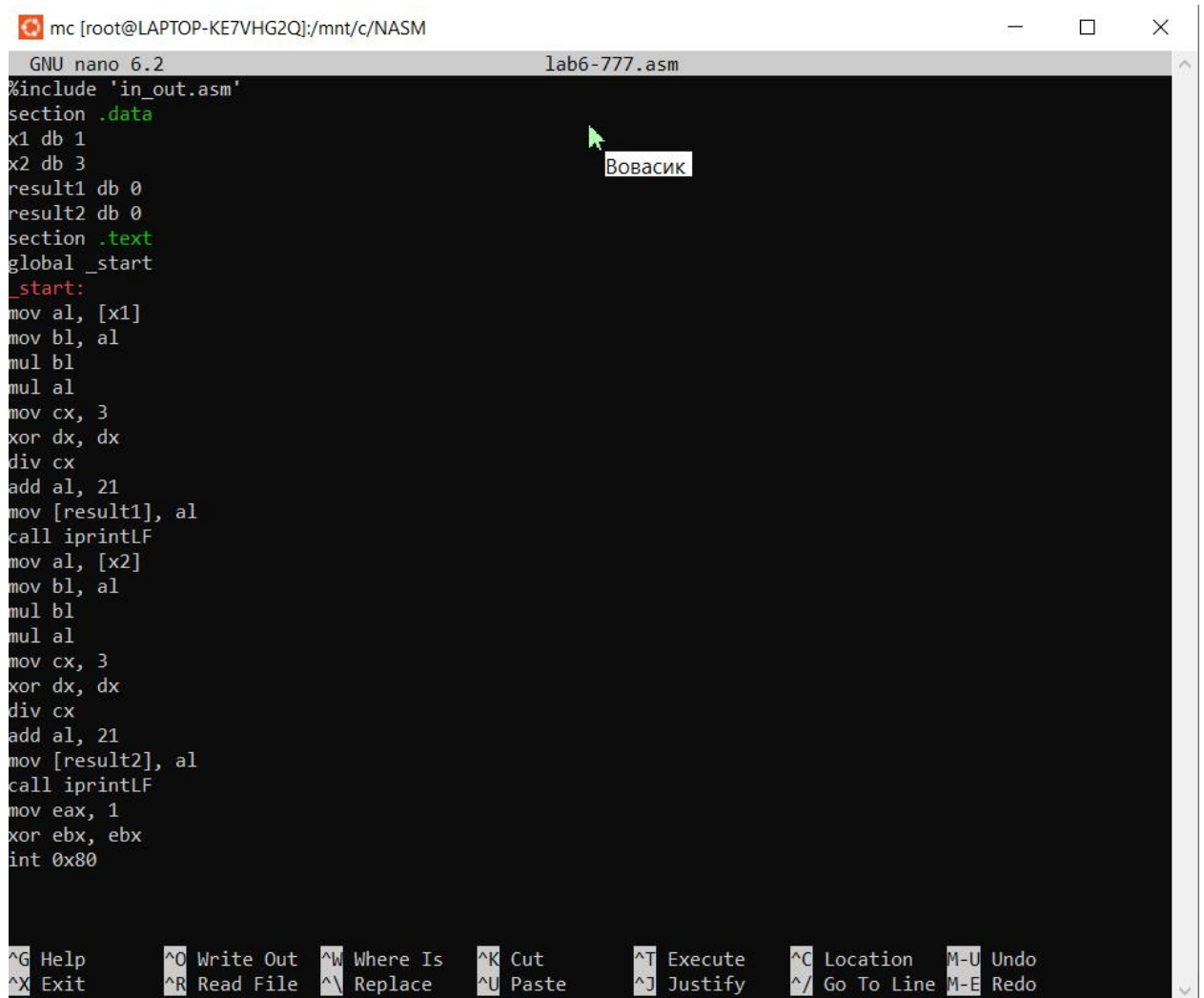
```
mov eax,edx
```

```
call iprintLF
```

Выполнение самостоятельной работы.

Написала программу вычисления выражения $y = f(x)$.

(Программа должна выводить выражение для вычисления, выводить запрос на ввод значения x , вычислять заданное выражение в зависимости от введенного x , выводить результат вычислений.)



```
mc [root@LAPTOP-KE7VHG2Q]:/mnt/c/NASM
GNU nano 6.2 lab6-777.asm
%include 'in_out.asm'
section .data
x1 db 1
x2 db 3
result1 db 0
result2 db 0
section .text
global _start
_start:
mov al, [x1]
mov bl, al
mul bl
mul al
mov cx, 3
xor dx, dx
div cx
add al, 21
mov [result1], al
call iprintLF
mov al, [x2]
mov bl, al
mul bl
mul al
mov cx, 3
xor dx, dx
div cx
add al, 21
mov [result2], al
call iprintLF
mov eax, 1
xor ebx, ebx
int 0x80

^G Help      ^O Write Out ^W Where Is  ^K Cut       ^T Execute   ^C Location  M-U Undo
^X Exit      ^R Read File ^\ Replace   ^U Paste     ^J Justify   ^_ Go To Line M-E Redo
```

Вид функции $f(x)$ выбрала из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы.

20	$x^3 \cdot \frac{1}{3} + 21$	1	3
----	------------------------------	---	---

Затем создала исполняемый файл и проверьте его работу для значений $x1$ и $x2$ из 6.3.

$x1 = 1$, ответ = 21 и $x2 = 3$, ответ = 48

```
root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab06# ld -m elf_i386 -o lab6-7 lab6-777.o
root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab06# ./lab6-7
21
48
root@LAPTOP-KE7VHG2Q:~/work/arch-pc/lab06#
```

Вывод.

Я узнала и ознакомилась с арифметическими инструкциями языка ассе

мблера NASM. Решая задания, я поняла, как работать с арифметическими действиями.

Список литературы

1. GDB: The GNU Project Debugger. — URL: <https://www.gnu.org/software/gdb/>.
2. GNU Bash Manual. — 2016. — URL: <https://www.gnu.org/software/bash/manual/>.
3. Midnight Commander Development Center. — 2021. — URL: <https://midnight-commander.org/>.
4. NASM Assembly Language Tutorials. — 2021. — URL: <https://asmtutor.com/>.
5. Newham C. Learning the bash Shell: Unix Shell Programming. — O'Reilly Media, 2005. — 354 с. — (In a Nutshell). — ISBN 0596009658. — URL: <http://www.amazon.com/Learningbash-Shell-Programming-Nutshell/dp/0596009658>.
6. Robbins A. Bash Pocket Reference. — O'Reilly Media, 2016. — 156 с. — ISBN 978-1491941591.
7. The NASM documentation. — 2021. — URL: <https://www.nasm.us/docs.php>.
8. Zarrelli G. Mastering Bash. — Packt Publishing, 2017. — 502 с. — ISBN 9781784396879.
9. Колдаев В. Д., Лупин С. А. Архитектура ЭВМ. — М. : Форум, 2018.
10. Куляс О. Л., Никитин К. А. Курс программирования на ASSEMBLER. — М. : Солон-Пресс, 2017.
11. Новожилов О. П. Архитектура ЭВМ и систем. — М. : Юрайт, 2016.
12. Расширенный ассемблер: NASM. — 2021. — URL: <https://www.opennet.ru/docs/RUS/nasm/>.
13. Робачевский А., Немнюгин С., Стесик О. Операционная система UNIX. — 2-е изд. — БХВПетербург, 2010. — 656 с. — ISBN 978-5-94157-538-1.
14. Столяров А. Программирование на языке ассемблера NASM для ОС Unix. — 2-е изд. — М. : МАКС Пресс, 2011. — URL: http://www.stolyarov.info/books/asm_unix.
15. Таненбаум Э. Архитектура компьютера. — 6-е изд. — СПб. : Питер, 2013. — 874 с. — (Классика Computer Science).
16. Таненбаум Э., Бос Х. Современные операционные системы. — 4-е изд. — СПб. : Питер, 2015. — 1120 с. — (Классика Computer Science)

