

Отчет по лабораторной работе №6

Простейший вариант

Сахно Алёна Юрьевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
5	Ответы на следующие вопросы:	17
6	Задание для самостоятельной работы	18
7	Выводы	19
	Список литературы	20

Список иллюстраций

4.1	Создание файла	9
4.2	Программа вывода значения регистра еах	10
4.3	Запуск исполняемого файла	11
4.4	Запуск исполняемого файла	12
4.5	Программа вывода значения регистра еах	13
4.6	Отличия функций	13
4.7	Программа вычисления выражения	14
4.8	Результат	15
4.9	Программа вычисления вычисления варианта задания по номеру студенческого билета	16
4.10	Программа вычисления вычисления варианта задания по номеру студенческого билета	16
6.1	Программа вычисления для $x_1=1$ и $x_2=4$	18

Список таблиц

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

2 Задание

1. Теоретическая часть
2. Порядок выполнения работы
3. Ответы на вопросы
4. Задание для самостоятельной работы
5. Вывод

3 Теоретическое введение

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.

Например, определим переменную `intg` DD 3 – это означает, что задается область памяти размером 4 байта, адрес которой обозначен меткой `intg`. В таком случае, команда

```
mov eax,[intg]
```

копирует из памяти по адресу `intg` данные в регистр `eax`. В свою очередь команда

```
mov [intg],eax
```

запишет в память по адресу `intg` данные из регистра `eax`. Также рассмотрим команду

```
mov eax,intg
```

В этом случае в регистр `eax` запишется адрес `intg`. Допустим, для `intg` выделена память начиная с ячейки с адресом `0x600144`, тогда команда `mov eax,intg` аналогична команде

```
mov eax,0x600144
```

 – т.е. эта команда запишет в регистр `eax` число `0x600144`

4 Выполнение лабораторной работы

1. Создайте каталог для программ лабораторной работы № 6, перейдите в него и создайте файл lab6-1.asm:

```
mkdir ~/work/arch-pc/lab06
```

```
cd ~/work/arch-pc/lab06
```

```
touch lab6-1.asm
```

(рис.1 4.1).

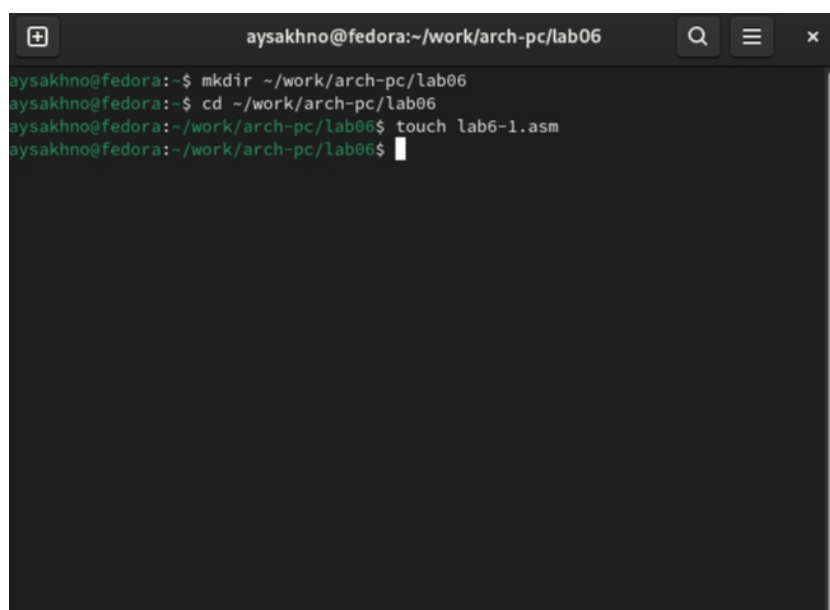
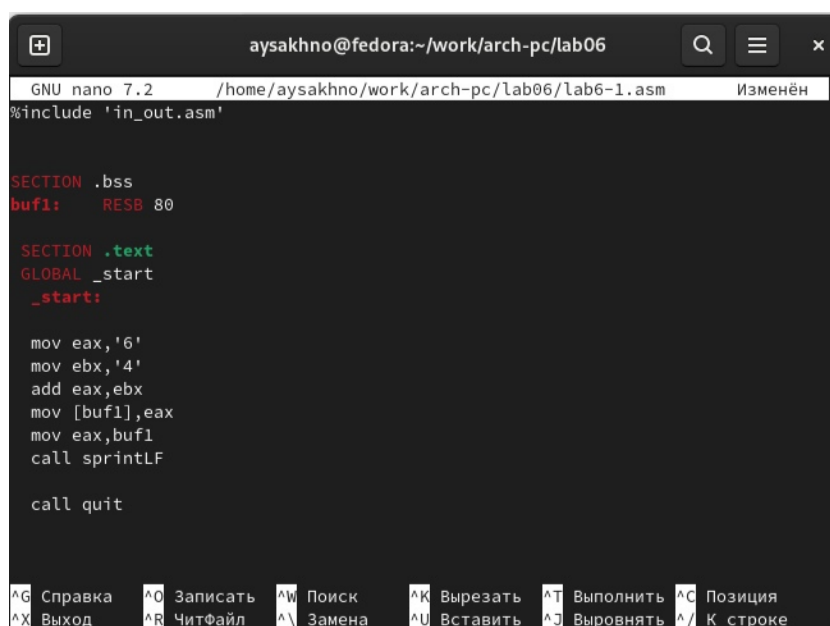
A screenshot of a terminal window with a dark background. The window title is 'aysakhno@fedora:~/work/arch-pc/lab06'. The terminal shows the following commands and their outputs: 'aysakhno@fedora:~\$ mkdir ~/work/arch-pc/lab06', 'aysakhno@fedora:~\$ cd ~/work/arch-pc/lab06', 'aysakhno@fedora:~/work/arch-pc/lab06\$ touch lab6-1.asm', and 'aysakhno@fedora:~/work/arch-pc/lab06\$' followed by a cursor. The window has standard Linux window controls (minimize, maximize, close) in the top right corner.

Рис. 4.1: Создание файла

2. Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения записанные в регистрах.

Введите в файл lab6-1.asm текст программы из листинга 6.1. В данной программе в регистр `eax` записывается символ 6 (`mov eax,'6'`), в регистр `ebx` символ 4 (`mov ebx,'4'`). Далее к значению в регистре `eax` прибавляем значение регистра `ebx` (`add eax,ebx`, результат сложения запишется в регистр `eax`). Далее выводим результат. Так как для работы функции `sprintLF` в регистр `eax` должен быть записан адрес, необходимо использовать дополнительную переменную. Для этого запишем значение регистра `eax` в переменную `buf1` (`mov [buf1],eax`), а затем запишем адрес переменной `buf1` в регистр `eax` (`mov eax,buf1`) и вызовем функцию `sprintLF`.

(рис.2 4.2).



```
GNU nano 7.2 /home/aysakhno/work/arch-pc/lab06/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 sprintLF

call quit
```

Рис. 4.2: Программа вывода значения регистра `eax`

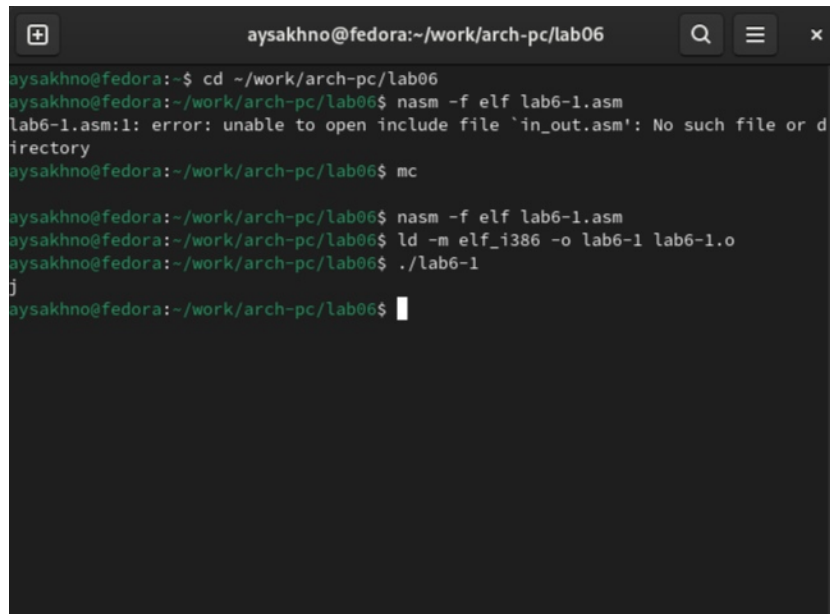
Создайте исполняемый файл и запустите его.

```
nasm -f elf lab6-1.asm
```

```
ld -m elf_i386 -o lab6-1 lab6-1.o
```

```
./lab6-1
```

(рис.3 4.3).

A terminal window titled 'aysakhno@fedora:~/work/arch-pc/lab06'. The terminal shows the following commands and output:

```
aysakhno@fedora:~$ cd ~/work/arch-pc/lab06
aysakhno@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
lab6-1.asm:1: error: unable to open include file 'in_out.asm': No such file or directory
aysakhno@fedora:~/work/arch-pc/lab06$ mc
aysakhno@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
aysakhno@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
aysakhno@fedora:~/work/arch-pc/lab06$ ./lab6-1
j
aysakhno@fedora:~/work/arch-pc/lab06$
```

Рис. 4.3: Запуск исполняемого файла

3. Далее изменим текст программы и вместо символов, запишем в регистры числа. Исправьте текст программы (Листинг 6.1) следующим образом: замените строки

```
mov eax,'6' mov ebx,'4'
```

на строки

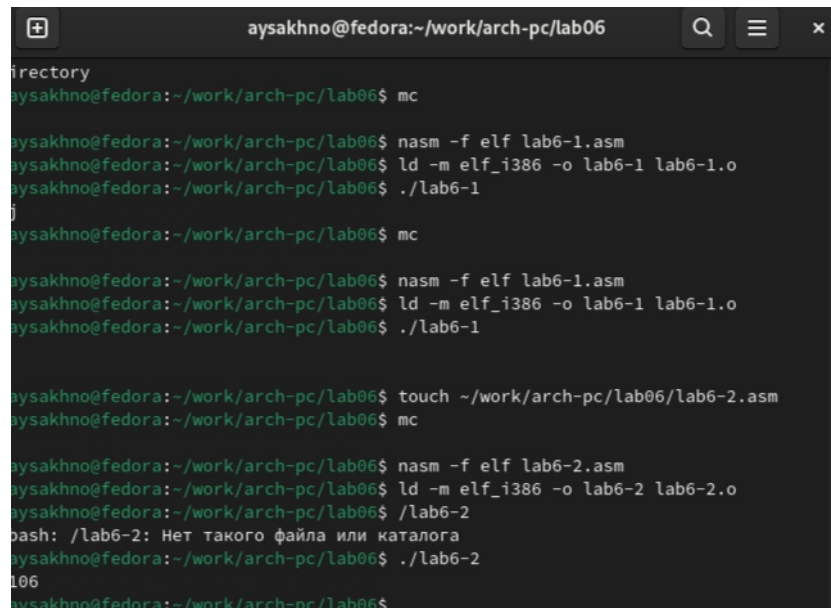
```
mov eax,6 mov ebx,4
```

Создайте исполняемый файл и запустите его. Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае выводится символ с кодом 10. Пользуясь таблицей ASCII определите какому символу соответствует код 10. Отображается ли этот символ при выводе на экран?

4. Как отмечалось выше, для работы с числами в файле in_out.asm реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Преобразуем текст программы из Листинга 6.1 с использованием этих функций. Создайте файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и введите в него текст программы из листинга 6.2.

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

(рис.4 4.4).



```
aysakhno@fedora:~/work/arch-pc/lab06$ mc
aysakhno@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
aysakhno@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
aysakhno@fedora:~/work/arch-pc/lab06$ ./lab6-1
106
aysakhno@fedora:~/work/arch-pc/lab06$ mc
aysakhno@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
aysakhno@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
aysakhno@fedora:~/work/arch-pc/lab06$ ./lab6-2
bash: ./lab6-2: Нет такого файла или каталога
aysakhno@fedora:~/work/arch-pc/lab06$ ./lab6-2
106
aysakhno@fedora:~/work/arch-pc/lab06$
```

Рис. 4.4: Запуск исполняемого файла

```
nasm -f elf lab6-2.asm
```

```
ld -m elf_i386 -o lab6-2 lab6-2.o
```

```
./lab6-2
```

В результате работы программы мы получим число 106. В данном случае, как и в первом, команда `add` складывает коды символов '6' и '4' ($54+52=106$). Однако, в отличие от программы из листинга 6.1, функция `iprintLF` позволяет вывести число, а не символ, кодом которого является это число.

5. Аналогично предыдущему примеру изменим символы на числа. Замените строки

```
mov eax,'6' mov ebx,'4'
```

на строки

```
mov eax,6 mov ebx,4
```

(рис.5 4.4).

```

GNU nano 7.2 /home/aysakhno/work/arch-pc/lab06/lab6-2.asm
#include 'in_out.asm'

SECTION .text
GLOBAL _start
_start:

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

call quit
  
```

Прочитано 12 строк

Справка Записать Поиск Вырезать Выполнить Позиция
 Выход ЧитФайл Замена Вставить Выводить К строке

Рис. 4.5: Программа вывода значения регистра eax

Создайте исполняемый файл и запустите его. Какой результат будет получен при исполнении программы? Замените функцию `iprintLF` на `iprint`. Создайте исполняемый файл и запустите его. Чем отличается вывод функций `iprintLF` и `iprint`?

(рис.6 4.6).

```

aysakhno@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
aysakhno@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
aysakhno@fedora:~/work/arch-pc/lab06$ ./lab6-2
10
aysakhno@fedora:~/work/arch-pc/lab06$ mc
aysakhno@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
aysakhno@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
aysakhno@fedora:~/work/arch-pc/lab06$ ./lab6-2
10aysakhno@fedora:~/work/arch-pc/lab06$
  
```

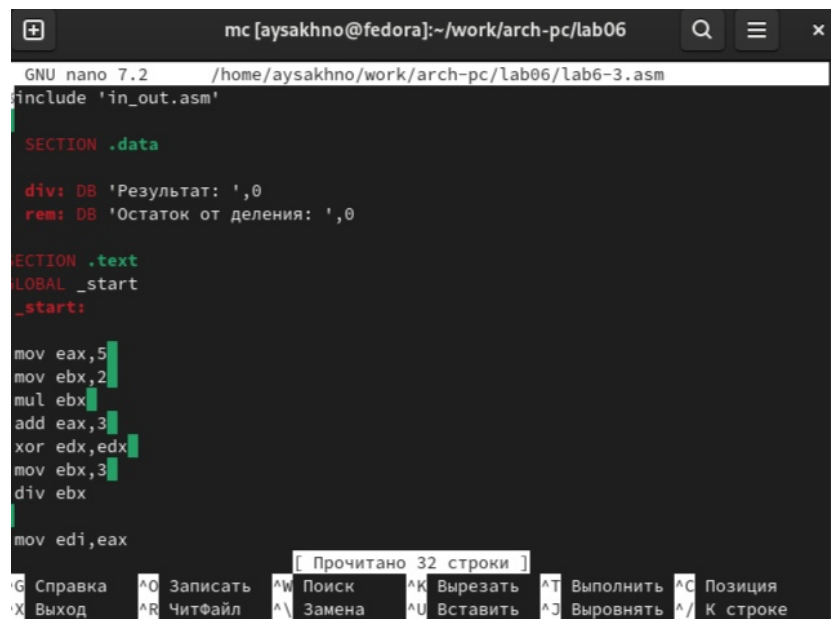
Рис. 4.6: Отличия функций

- В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения $\frac{(5 \cdot 2 + 3)}{3}$.

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

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

(рис.7 4.7).



```
GNU nano 7.2 /home/aysakhno/work/arch-pc/lab06/lab6-3.asm
#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

[ Прочитано 32 строки ]
^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^_ Выровнять ^/ К строке
```

Рис. 4.7: Программа вычисления выражения

Создайте исполняемый файл и запустите его. Результат работы программы должен быть следующим:

```
user@dk4n31:~$ ./lab6-3
```

```
Результат: 4 Остаток от деления: 1
```

```
user@dk4n31:~$
```

(рис.8 4.8).

```
aysakhno@fedora:~/work/arch-pc/lab06$ mc
aysakhno@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
aysakhno@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
aysakhno@fedora:~/work/arch-pc/lab06$ ./lab6-2
10
aysakhno@fedora:~/work/arch-pc/lab06$ mc
aysakhno@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
aysakhno@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
aysakhno@fedora:~/work/arch-pc/lab06$ ./lab6-2
10aysakhno@fedora:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-3.asm
aysakhno@fedora:~/work/arch-pc/lab06$ mc
aysakhno@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
aysakhno@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-3.o
aysakhno@fedora:~/work/arch-pc/lab06$ ./lab6-3
bash: ./lab6-3: Нет такого файла или каталога
aysakhno@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
aysakhno@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
aysakhno@fedora:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
aysakhno@fedora:~/work/arch-pc/lab06$
```

Рис. 4.8: Результат

7. В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета, работающую по следующему алгоритму:

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

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

(рис.9 4.9).

```

aysakhno@fedora:~/work/arch-pc/lab06
GNU nano 7.2 /home/aysakhno/work/arch-pc/lab06/variant.asm
%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 read

[ Прочитано 34 строки ]
^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выводить ^/_ К строке

```

Рис. 4.9: Программа вычисления вычисления варианта задания по номеру студенческого билета

Создайте исполняемый файл и запустите его. Проверьте результат работы программы вычислив номер варианта аналитически.

(рис.10 4.10).

```

aysakhno@fedora:~/work/arch-pc/lab06$ nasm -f elf variant.asm
aysakhno@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
aysakhno@fedora:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132243813
Ваш вариант: 14
aysakhno@fedora:~/work/arch-pc/lab06$

```

Рис. 4.10: Программа вычисления вычисления варианта задания по номеру студенческого билета

5 Ответы на следующие вопросы:

1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения 'Ваш вариант: '? Ответ: `mov eax,rem call sprint mov eax,edx call iprintLF call quit`

2. Для чего используются следующие инструкции?

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

Ответ: 1- Адрес строки x 2- запись длины вводимого сообщения 3 - вызов подпрограммы ввода сообщения

3. Для чего используется инструкция "call atoi"?

Ответ: для ASCII кода в число

4. Какие строки листинга 6.4 отвечают за вычисления варианта?

Ответ: `inc edx`

5. В какой регистр записывается остаток от деления при выполнении инструкции "div ebx"? Ответ: `div`

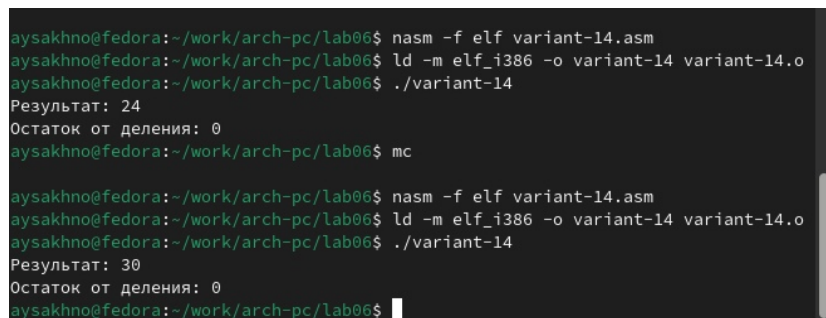
6. Для чего используется инструкция "inc edx"? Ответ: команда `inc edx` уменьшает значение регистра `edx` на 1

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

Ответ: `call sprint mov eax,edx call iprintLF call quit`

6 Задание для самостоятельной работы

(рис.11 6.1).



```
aysakhno@fedora:~/work/arch-pc/lab06$ nasm -f elf variant-14.asm
aysakhno@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant-14 variant-14.o
aysakhno@fedora:~/work/arch-pc/lab06$ ./variant-14
Результат: 24
Остаток от деления: 0
aysakhno@fedora:~/work/arch-pc/lab06$ mc

aysakhno@fedora:~/work/arch-pc/lab06$ nasm -f elf variant-14.asm
aysakhno@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant-14 variant-14.o
aysakhno@fedora:~/work/arch-pc/lab06$ ./variant-14
Результат: 30
Остаток от деления: 0
aysakhno@fedora:~/work/arch-pc/lab06$
```

Рис. 6.1: Программа вычисления для $x1=1$ и $x2=4$

7 Выводы

Я освоила арифметические инструкции языка ассемблера NASM.

Список литературы

:::{#refs}:: https://esystem.rudn.ru/pluginfile.php/2089086/mod_resource/content/0/%D0%9B%D