

Лабораторная работа №6

Архитектура вычислительных систем

Горчаков Егор Алексеевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
5	Ответы на вопросы:	19
6	Выводы	20
	Список литературы	21

Список иллюстраций

4.1	61.png	9
4.2	62.png	10
4.3	63.png	10
4.4	64.png	11
4.5	65.png	11
4.6	66.png	12
4.7	67.png	12
4.8	68.png	12
4.9	69.png	13
4.10	610.png	14
4.11	611.png	15
4.12	612.png	16
4.13	613.png	17
4.14	614png	17
4.15	615png	17
4.16	616png	18
4.17	617png	18

Список таблиц

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

2 Задание

Написать программу вычисления выражения. Программа должна выводить выражение для вычисления, выводить запрос на ввод значения x , вычислять заданное выражение в зависимости от введенного x , выводить результат вычислений. Вид функции выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. Создать исполняемый файл и проверить его работу для значений из 6.3.

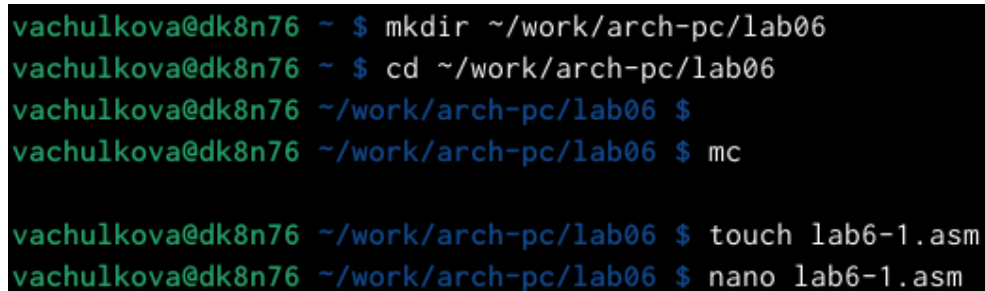
3 Теоретическое введение

1. Адресация в NASM 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.
2. Арифметические операции в NASM Схема команды целочисленного сложения `add` (от англ. addition - добавление) выполняет сложение двух операндов и записывает результат по адресу первого операнда. Команда `add` работает как с числами со знаком, так и без знака.
3. Целочисленное вычитание `sub` Команда целочисленного вычитания `sub` (от англ. subtraction – вычитание) работает аналогично команде `add`.
4. Команды инкремента и декремента Довольно часто при написании программ встречается операция прибавления или вычитания единицы. Прибавление единицы называется инкрементом, а вычитание — декрементом. Для этих операций существуют специальные команды: `inc` (от англ. increment) и `dec` (от англ. decrement), которые увеличивают и уменьшают на 1 свой операнд.
5. Команда изменения знака операнда `neg` Команда рассматривает свой операнд как число со знаком и меняет знак операнда на противоположный. Операндом может быть регистр или ячейка памяти любого размера.

6. Команды умножения `mul` и `imul` Умножение и деление, в отличие от сложения и вычитания, для знаковых и беззнаковых чисел производиться по-разному, поэтому существуют различные команды. Для беззнакового умножения используется команда `mul` (от англ. `multiply` – умножение). Для знакового умножения используется команда `imul`.
7. Команды деления `div` и `idiv` Для деления, как и для умножения, существует 2 команды `div` (от англ. `divide` - деление) и `idiv`. Для беззнакового умножения используется команда `div`. Для знакового умножения используется команда `idiv`.

4 Выполнение лабораторной работы

1. Создадим директорию для лабораторной работы №6.
2. Перейдем в нее и создадим файл lab6-1.asm.



```
vachulkova@dk8n76 ~ $ mkdir ~/work/arch-pc/lab06
vachulkova@dk8n76 ~ $ cd ~/work/arch-pc/lab06
vachulkova@dk8n76 ~/work/arch-pc/lab06 $
vachulkova@dk8n76 ~/work/arch-pc/lab06 $ mc

vachulkova@dk8n76 ~/work/arch-pc/lab06 $ touch lab6-1.asm
vachulkova@dk8n76 ~/work/arch-pc/lab06 $ nano lab6-1.asm
```

Рис. 4.1: 61.png

3. Введем в файл lab6-1.asm текст программы из листинга 6.1.

```
GNU nano 6.3 lab6-1.asm Изменён
#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 _exit
```

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^C Позиция

Рис. 4.2: 62.png

4. Создадим копию файла in_out.asm в каталоге.

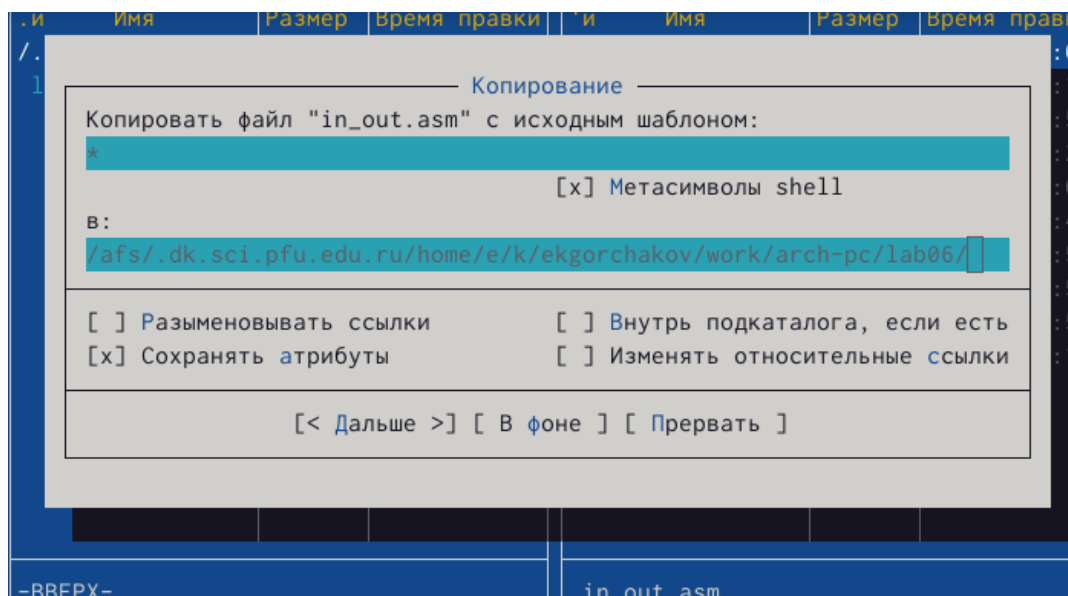
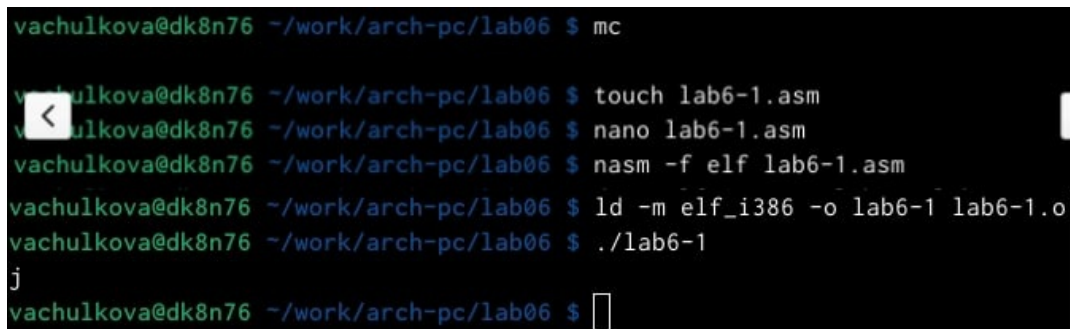


Рис. 4.3: 63.png

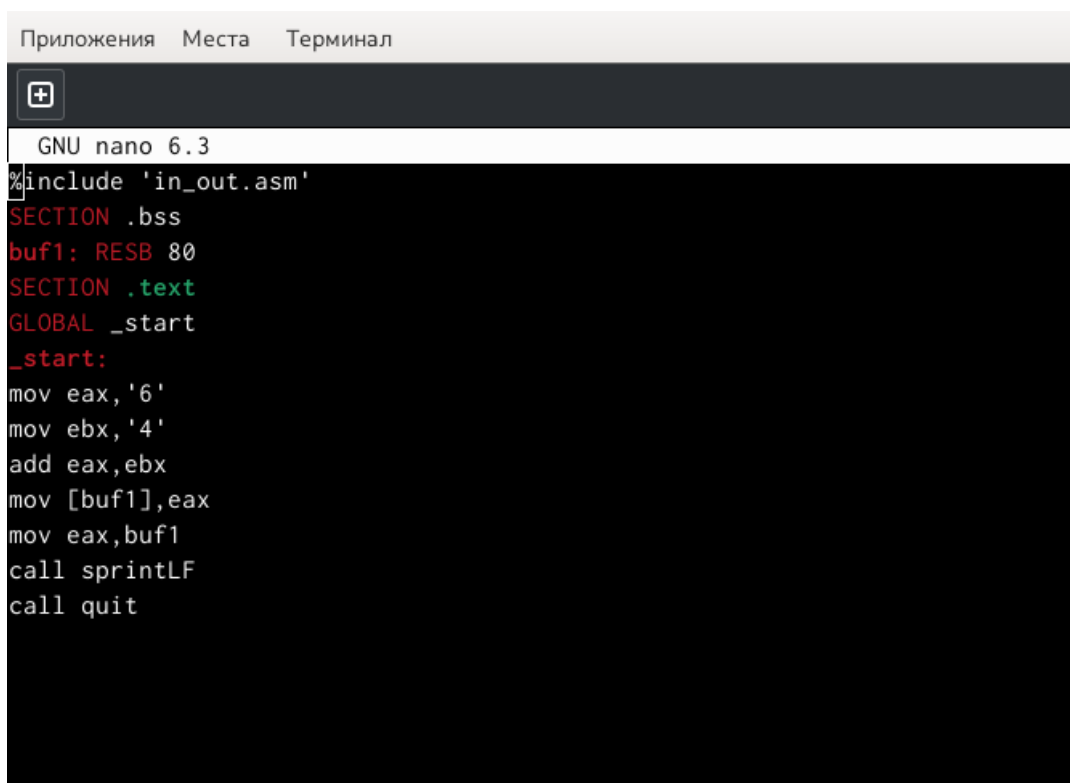
5. Создадим исполняемый файл и запустим его.



```
vachulkova@dk8n76 ~/work/arch-pc/lab06 $ mc
vachulkova@dk8n76 ~/work/arch-pc/lab06 $ touch lab6-1.asm
vachulkova@dk8n76 ~/work/arch-pc/lab06 $ nano lab6-1.asm
vachulkova@dk8n76 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
vachulkova@dk8n76 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
vachulkova@dk8n76 ~/work/arch-pc/lab06 $ ./lab6-1
j
vachulkova@dk8n76 ~/work/arch-pc/lab06 $
```

Рис. 4.4: 64.png

6. Далее изменим текст программы и вместо символов, запишем в регистры числа. Исправим текст программы.



```
Приложения  Места  Терминал
GNU nano 6.3
%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. 4.5: 65.png

7. Создадим исполняемый файл и запустим его (6-1).

```

vachulkova@dk5n51 ~/work/arch-pc/lab06 $ nano lab6-1.asm
vachulkova@dk5n51 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
vachulkova@dk5n51 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
vachulkova@dk5n51 ~/work/arch-pc/lab06 $ ./lab6-1

vachulkova@dk5n51 ~/work/arch-pc/lab06 $ 

```

Рис. 4.6: 66.png

8. Создадим файл lab6-2.asm в каталоге. Введем в него текст программы из листинга 6.2 и запустим его.

```

vachulkova@dk8n81 ~/work/arch-pc/lab06 $ touch lab6-2.asm
vachulkova@dk8n81 ~/work/arch-pc/lab06 $ nano lab6-2.asm
vachulkova@dk8n81 ~/work/arch-pc/lab06 $ 
vachulkova@dk8n81 ~/work/arch-pc/lab06 $ nano lab6-2.asm
vachulkova@dk8n81 ~/work/arch-pc/lab06 $ gedit lab6-2.asm
vachulkova@dk8n81 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
vachulkova@dk8n81 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
vachulkova@dk8n81 ~/work/arch-pc/lab06 $ ./lab6-2
106

```

Рис. 4.7: 67.png

```

GNU nano 6.3 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

```

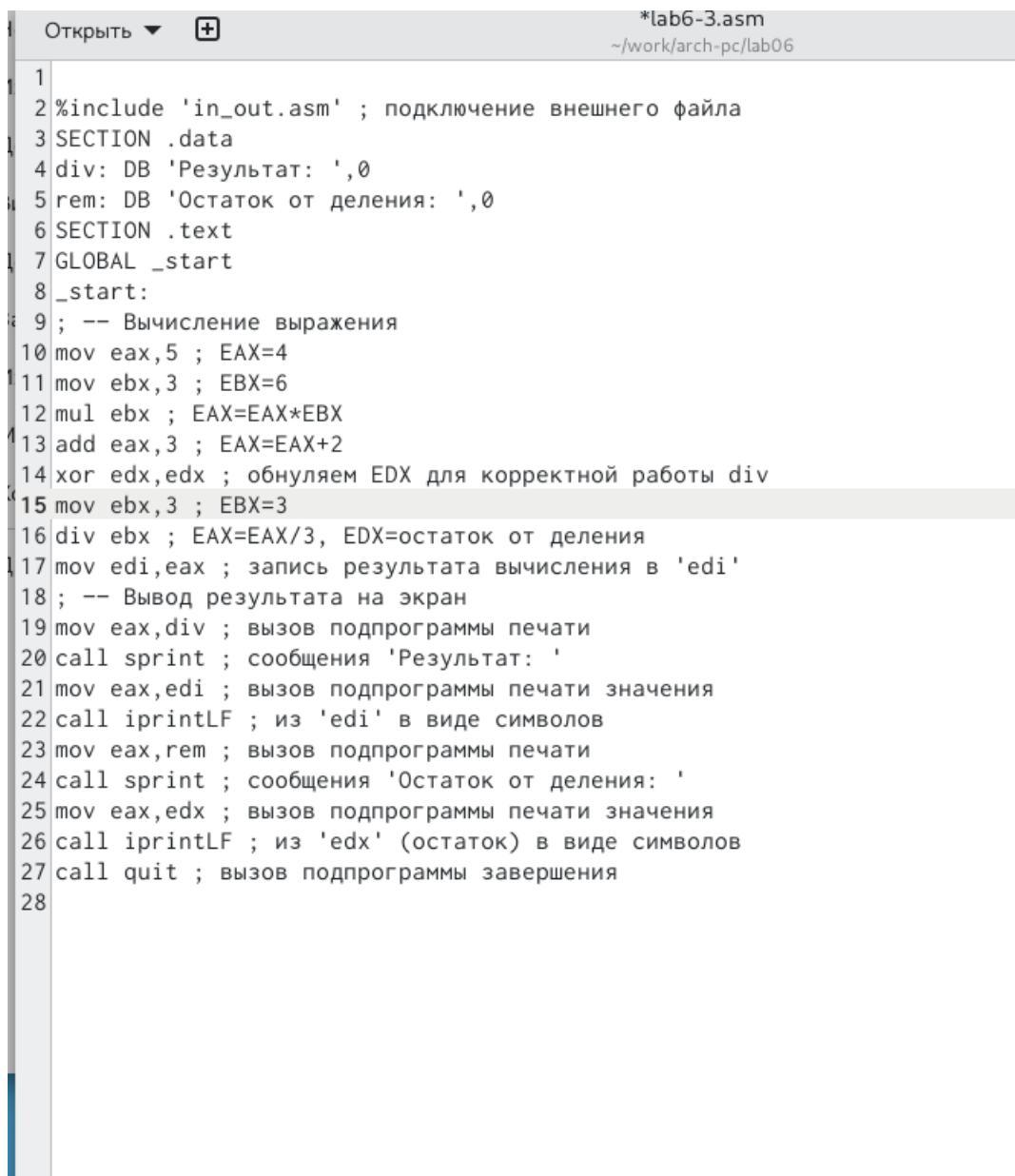
Рис. 4.8: 68.png

9. Изменим символы на числа в lab6-2. Создадим исполняемый файл и запустим его.

```
vachulkova@dk8n81 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
vachulkova@dk8n81 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
vachulkova@dk8n81 ~/work/arch-pc/lab06 $ ./lab6-2
10
```

Рис. 4.9: 69.png

10. Создадим файл lab6-3.asm в каталоге. Введем в файл lab6-3.asm текст программы из листинга 6.3



```
1
2 %include 'in_out.asm' ; подключение внешнего файла
3 SECTION .data
4 div: DB 'Результат: ',0
5 rem: DB 'Остаток от деления: ',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 ; -- Вычисление выражения
10 mov eax,5 ; EAX=4
11 mov ebx,3 ; EBX=6
12 mul ebx ; EAX=EAX*EBX
13 add eax,3 ; EAX=EAX+2
14 xor edx,edx ; обнуляем EDX для корректной работы div
15 mov ebx,3 ; EBX=3
16 div ebx ; EAX=EAX/3, EDX=остаток от деления
17 mov edi,eax ; запись результата вычисления в 'edi'
18 ; -- Вывод результата на экран
19 mov eax,div ; вызов подпрограммы печати
20 call sprint ; сообщения 'Результат: '
21 mov eax,edi ; вызов подпрограммы печати значения
22 call iprintLF ; из 'edi' в виде символов
23 mov eax,rem ; вызов подпрограммы печати
24 call sprint ; сообщения 'Остаток от деления: '
25 mov eax,edx ; вызов подпрограммы печати значения
26 call iprintLF ; из 'edx' (остаток) в виде символов
27 call quit ; вызов подпрограммы завершения
28
```

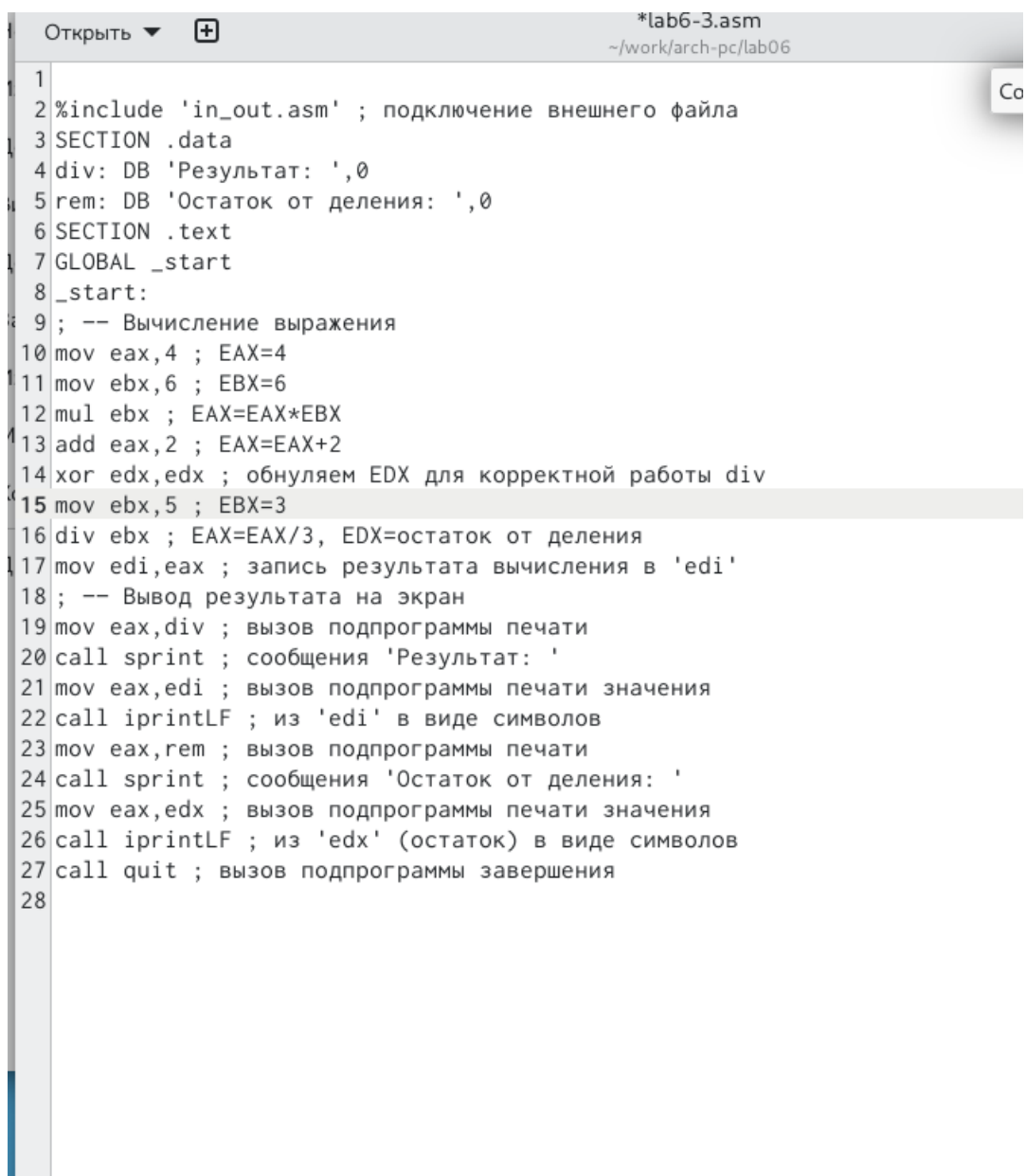
Рис. 4.10: 610.png

11. Создадим исполняемый файл и запустим его.

```
vachulkova@dk8n81 ~/work/arch-pc/lab06 $ gedit lab6-3.asm
vachulkova@dk8n81 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
vachulkova@dk8n81 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
vachulkova@dk8n81 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
vachulkova@dk8n81 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
vachulkova@dk8n81 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
```

Рис. 4.11: 611.png

12. Введем в файл lab6-3 программу вычисления выражения .



```
1
2 %include 'in_out.asm' ; подключение внешнего файла
3 SECTION .data
4 div: DB 'Результат: ',0
5 rem: DB 'Остаток от деления: ',0
6 SECTION .text
7 GLOBAL _start
8 _start:
9 ; -- Вычисление выражения
10 mov eax,4 ; EAX=4
11 mov ebx,6 ; EBX=6
12 mul ebx ; EAX=EAX*EBX
13 add eax,2 ; EAX=EAX+2
14 xor edx,edx ; обнуляем EDX для корректной работы div
15 mov ebx,5 ; EBX=3
16 div ebx ; EAX=EAX/3, EDX=остаток от деления
17 mov edi,eax ; запись результата вычисления в 'edi'
18 ; -- Вывод результата на экран
19 mov eax,div ; вызов подпрограммы печати
20 call sprint ; сообщения 'Результат: '
21 mov eax,edi ; вызов подпрограммы печати значения
22 call iprintLF ; из 'edi' в виде символов
23 mov eax,rem ; вызов подпрограммы печати
24 call sprint ; сообщения 'Остаток от деления: '
25 mov eax,edx ; вызов подпрограммы печати значения
26 call iprintLF ; из 'edx' (остаток) в виде символов
27 call quit ; вызов подпрограммы завершения
28
```

Рис. 4.12: 612.png

13. Создадим исполняемый файл и запустим его для вычисления выражения.


```

vachulkova@dk8n81 ~/work/arch-pc/lab06 $ gedit lab6-3.asm
vachulkova@dk8n81 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
vachulkova@dk8n81 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
vachulkova@dk8n81 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1

```

Рис. 4.13: 613.png

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

```

vachulkova@dk8n81 ~/work/arch-pc/lab06 $ touch variant.asm
vachulkova@dk8n81 ~/work/arch-pc/lab06 $ gedit variant.asm

```

Рис. 4.14: 614.png

15. Вводим номер студенческого и получаем вариант для выполнения задания

```

vachulkova@dk8n81 ~/work/arch-pc/lab06 $ ./variant
Введите No студенческого билета:
1132227115
Ваш вариант: 12
vachulkova@dk8n81 ~/work/arch-pc/lab06 $

```

Рис. 4.15: 615.png

16. Составляем программу для нашего варианта lab6-4 (Самостоятельная работа).

```

1 %include 'in_out.asm' ;
2 SECTION .data
3 div: DB 'Результат : ',0
4 rem: DB 'Введите переменную x: ',0
5 rem1: DB 'x будет ',0
6 SECTION .bss
7 x: RESB 80
8 SECTION .text
9 GLOBAL _start
10 _start:
11 ; -----Вычисление выражения
12 mov eax,rem
13 call sprintf
14 mov eax, rem1
15 call sprintf
16 mov ecx, x
17 mov edx,80
18 call sread
19 mov eax, x
20 call atoi
21 mov ebx,8
22 mul ebx
23 sub eax,6
24
25 xor edx,edx
26 mov ebx,2
27 div ebx
28 mov edi,eax
29 mov eax, div
30 call sprintf
31 mov eax, edi
32 call iprintLF
33 call quit

```

Рис. 4.16: 616png

17. Запускаем программу и вводим два числа из условия, убеждаемся что программа работает верно (Самостоятельная работа).

```

vachulkova@dk8n81 ~/work/arch-pc/lab06 $ nasm -f elf lab6-4.asm
vachulkova@dk8n81 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-4 lab6-4.o
vachulkova@dk8n81 ~/work/arch-pc/lab06 $ ./lab6-4
Введите переменную x:
x будет 1
Результат :1
vachulkova@dk8n81 ~/work/arch-pc/lab06 $ ./lab6-4
Введите переменную x:
x будет 5
Результат :17

```

Рис. 4.17: 617png

5 Ответы на вопросы:

1. `mov eax, x` и `rem call sprint`;
2. `mov ecx, x` - запись входной переменной в регистр `ecx`; `mov edx, 80` - запись размера переменной в регистр `edx`; `call sread` - вызов процедуры чтения данных;
3. `call atoi` - функция преобразующая ASCII код символа в целое число и записывающая результат в регистр `eax`;
4. `xor edx, edx` `mov ebx, 20` `div ebx`, `inc edx`;
5. `div ebx` - `ebx`;
6. `inc` - используется для увеличения операнда на единицу;
7. `mov eax, x` `rem call sprint` `mov eax, edx` `call iprintLF`.

6 Выводы

В ходе выполнения данной лабораторной работы были освоены арифметические инструкции языка ассемблера NASM.

Список литературы