

Лабораторная работа №6

Архитектура вычислительных систем

Бутерин Арсений Геворгович

Содержание

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

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	12
4.6	67.png	13
4.7	68.png	13
4.8	69.png	14
4.9	610.png	14
4.10	611.png	15
4.11	612.png	15
4.12	613png	16
4.13	614png	16

Список таблиц

1 Цель работы

Освоить арифметические инструкции 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 производятся 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. Создаём каталог для программ лабораторной работы No 7, перейдём в него и создаём файл lab6-1.asm

```
agbuterin@dk5n56 ~ $ mkdir ~/work/arch-pc/lab06
agbuterin@dk5n56 ~ $ cd ~/work/arch-pc/lab06
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ touch lab6-1.asm
agbuterin@dk5n56 ~/work/arch-pc/lab06 $
```

Рис. 4.1: 61.png

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

```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/a/g/ag
%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. 4.2: 62.png

3. Создаём копию файла in_out.asm в каталоге.

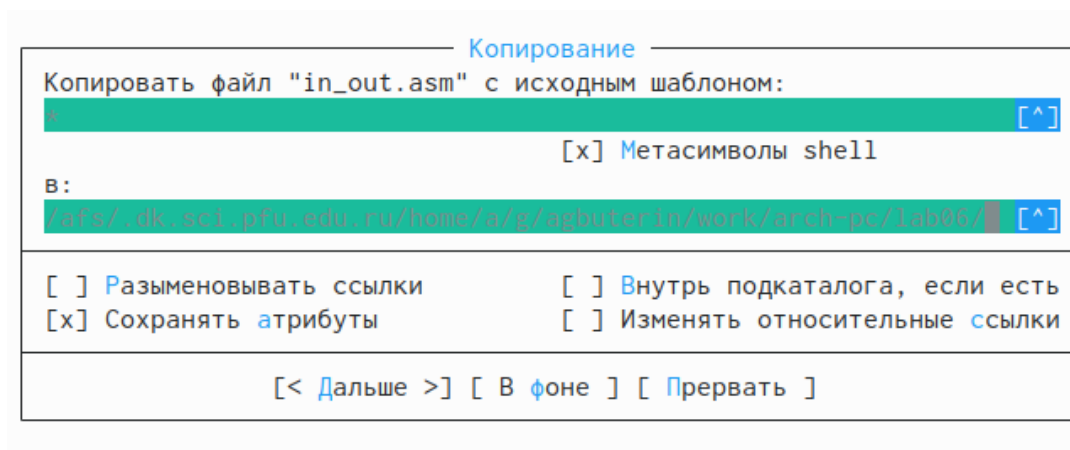


Рис. 4.3: 63.png

4. Создадим исполняемый файл и запустим его.

```
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ ld -m elf_I386 -o lab6-1 lab6-i.o
ld: не распознан режим эмуляции: elf_I386
Поддерживаемые эмуляции: elf_x86_64 elf32_x86_64 elf_i386 elf_iamcu
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-i.o
ld: невозможно найти lab6-i.o: Нет такого файла или каталога
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ ./lab6-1
j
agbuterin@dk5n56 ~/work/arch-pc/lab06 $
```

Рис. 4.4: 64.png

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

```
GNU nano 6.4 /afs/.dk.sci.pfu
%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

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

```
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ ./lab6-1

agbuterin@dk5n56 ~/work/arch-pc/lab06 $
```

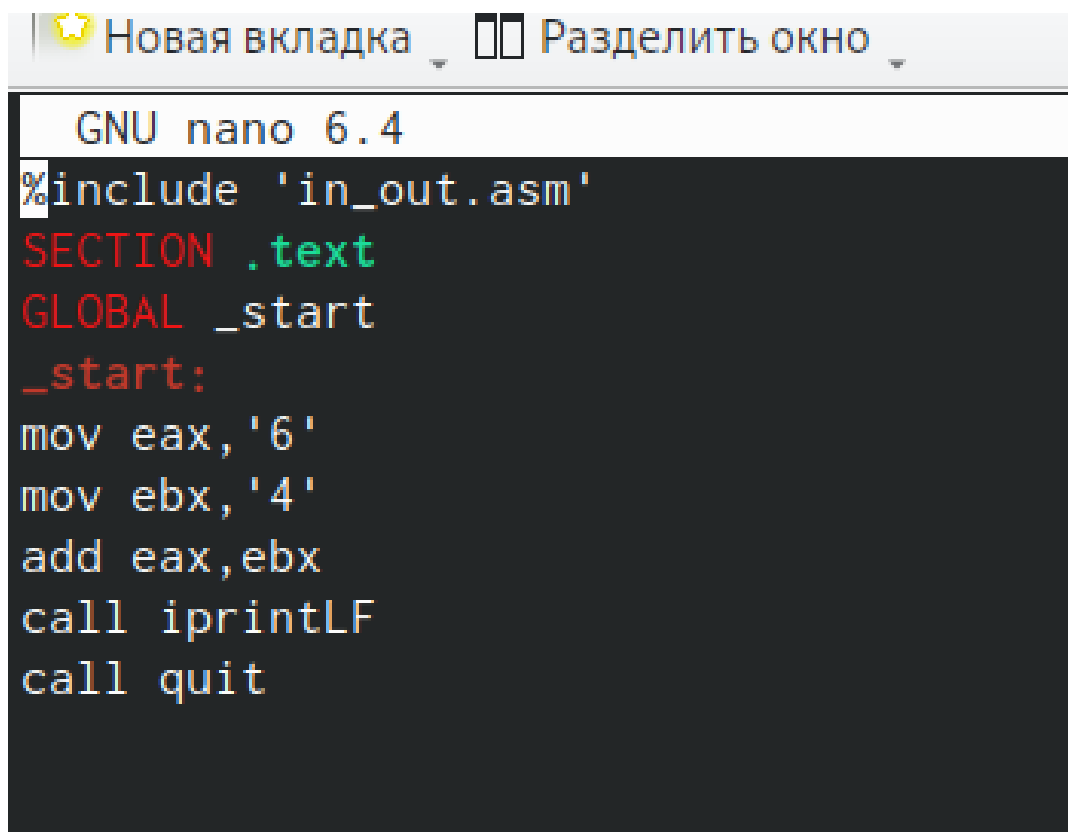
Рис. 4.6: 67.png

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

```
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ touch lab6-2.asm
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ nano lab6-2.asm
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ ./lab6-2
106
agbuterin@dk5n56 ~/work/arch-pc/lab06 $
```

Рис. 4.7: 68.png

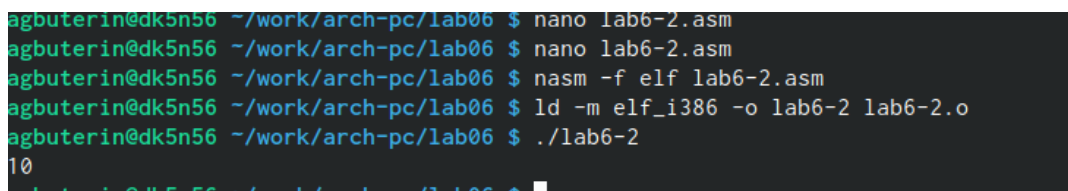
8. Изменим символы на числа в lab6-2. Создадим исполняемый файл и запустим его.



```
GNU nano 6.4
%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: 69.png

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



```
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ nano lab6-2.asm
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ nano lab6-2.asm
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ ./lab6-2
10
agbuterin@dk5n56 ~/work/arch-pc/lab06 $
```

Рис. 4.9: 610.png

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

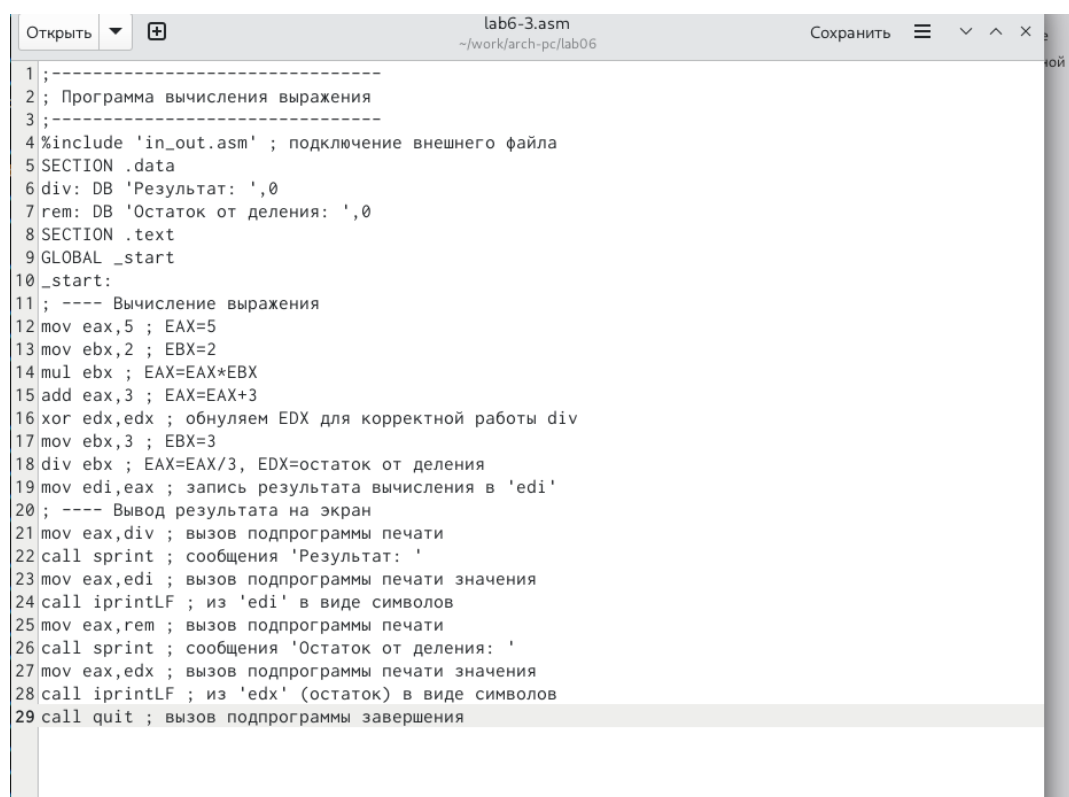
```

10
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ touch lab6-3.asm
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ gedit lab6-3.asm
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ █

```

Рис. 4.10: 611.png

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



```

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

```

Рис. 4.11: 612.png

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

14. Вводим номер студенческого и получаем вариант для выполнения задания

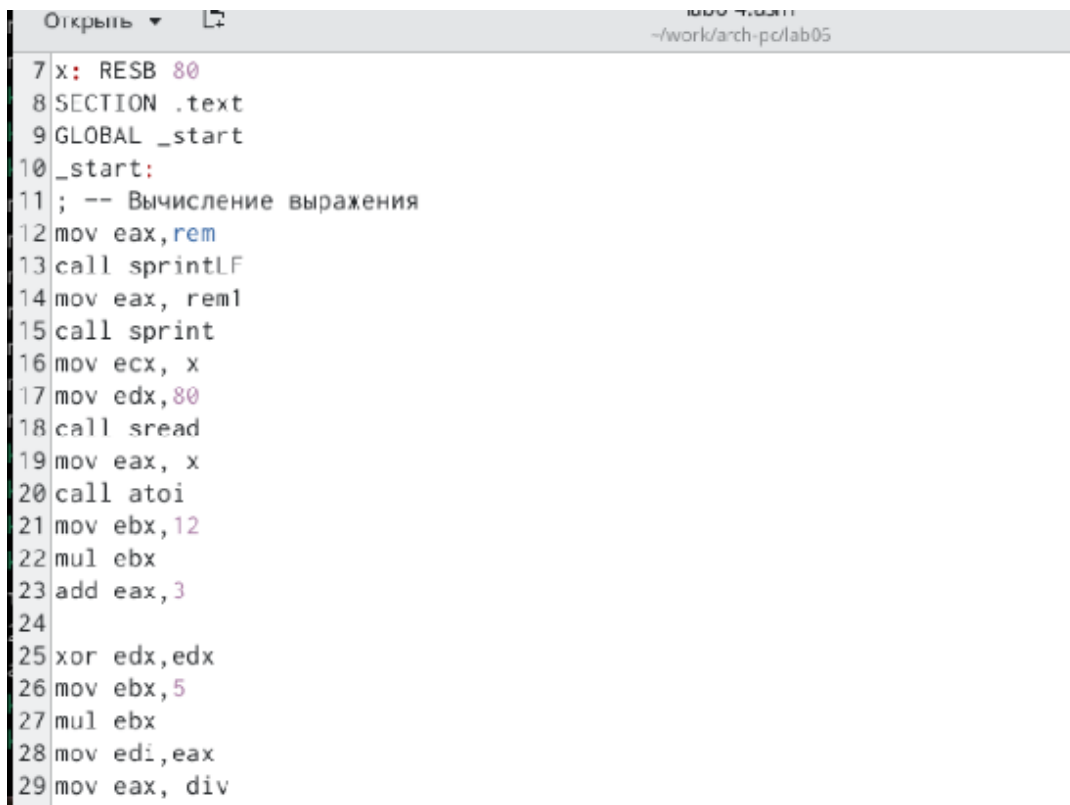
```

agbuterin@dk5n56 ~/work/arch-pc/lab06 $ gedit lab6-3.asm
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ touch variant.asm
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ gedit variant.asm
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
agbuterin@dk5n56 ~/work/arch-pc/lab06 $ ./variant
Введите No студенческого билета:
1132230800
Ваш вариант: 1
agbuterin@dk5n56 ~/work/arch-pc/lab06 $

```

Рис. 4.12: 613png

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



```

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, 12
22 mul ebx
23 add eax, 3
24
25 xor edx, edx
26 mov ebx, 5
27 mul ebx
28 mov edi, eax
29 mov eax, div

```

Рис. 4.13: 614png

16. Запускаем программу и вводим два числа из условия, убеждаемся что программа работает верно.

Введите переменную x:

x будет 1

Результат :75

16png

5 Ответы на вопросы:

1. строки листинга 7.4 отвечают за вывод на экран сообщения 'Ваш вариант:
mov eax 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 за-
писывающая 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. Следующие строки листинга отвечают за вывод на экран результата вычис-
лений mov eax, rem call sprint mov eax, edx call iprintLF.

6 Выводы

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

Список литературы