

Шаблон отчёта по лабораторной работе

Простейший вариант

Сокирка Анна Константиновна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
5	4.1 Символьные и численные данные в NASM	10
6	4.2 Выполнение арифметических операций в NASM	14
7	4.2.1 Ответы на вопросы по программе	16
8	4.3 Выполнение заданий для самостоятельной работы	17
9	Выводы	19
10	Список литературы	20

Список иллюстраций

5.1	Создание каталога	10
5.2	Ввод текста из листинга	10
5.3	Копирование файла	11
5.4	Запуск исполняемого файла	11
5.5	Изменение текста программы	11
5.6	Создание файла	12
5.7	Создание файла	12
5.8	Создание исполняемого файла	12
5.9	Изменение текста программы	13
5.10	Создание файла	13
5.11	Изменение текста программы	13
6.1	Создание файла	14
6.2	Создание и запуск файла	15
6.3	Проверка исполняемого файла	15
6.4	Запуск программы	15
8.1	Создание файла	17
8.2	Ввод текста программы	18
8.3	Запуск исполняемого файла	18
8.4	Проверка работы исполняемого файла	18

Список таблиц

1 Цель работы

Цель данной лабораторной работы - освоение арифметических инструкций языка ассемблера NASM.

2 Задание

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM
3. Выполнение заданий для самостоятельной работы

3 Теоретическое введение

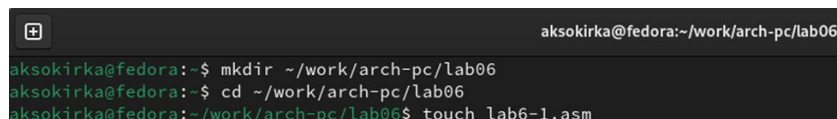
Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти. - Регистровая адресация – операнды хранятся в регистрах и в команде используются имена 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. Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. ASCII – сокращение от American Standard Code for Information Interchange (Американский стандартный код для обмена информацией). Согласно стандарту ASCII каждый символ кодируется одним байтом. Среди инструкций NASM нет такой, которая выводит числа (не в символьном виде). Поэтому, например, чтобы вывести число, надо предварительно преобразовать его цифры в ASCII-коды этих цифр и выводить на экран эти коды, а не само число. Если же выводить число на экран непосредственно, то экран воспримет его не как число, а как последовательность ASCII-символов – каждый байт числа будет воспринят как один ASCII-символ – и выведет на экран эти символы. Аналогичная ситуация происходит и при вводе данных с клавиатуры. Введенные данные будут представлять собой символы, что сделает невозможным получение корректного результата при выполнении над ними

арифметических операций. Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно.

4 Выполнение лабораторной работы

5 4.1 Символьные и численные данные в NASM

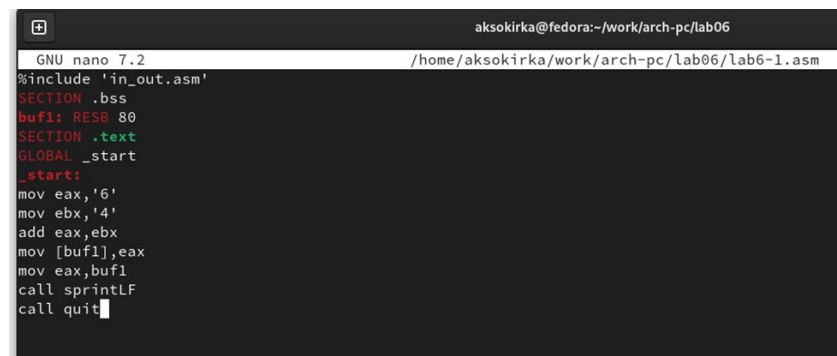
Создам каталог для программ лабораторной работы № 6, перейду в него и создам файл lab6-1.asm (рис. 5.1).



```
aksokirka@fedora:~/work/arch-pc/lab06
aksokirka@fedora:~$ mkdir ~/work/arch-pc/lab06
aksokirka@fedora:~$ cd ~/work/arch-pc/lab06
aksokirka@fedora:~/work/arch-pc/lab06$ touch lab6-1.asm
```

Рис. 5.1: Создание каталога

Введу в файл lab6-1.asm текст программы из листинга 6.1 (рис. 5.2).



```
GNU nano 7.2 /home/aksokirka/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 sprintf
call quit
```

Рис. 5.2: Ввод текста из листинга

Копирую файл in_out.asm в файл lab06 (рис. 5.3).

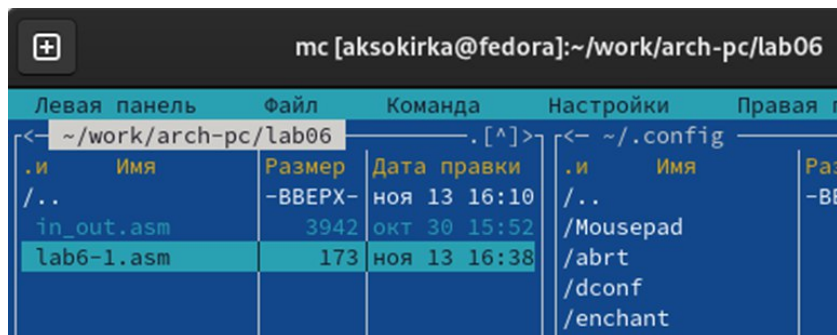


Рис. 5.3: Копирование файла

Создам исполняемый файл и запущу его (рис. 5.4).

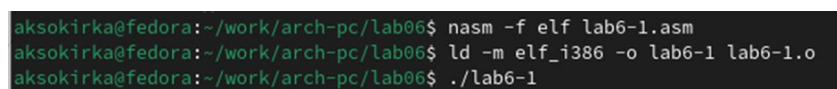


Рис. 5.4: Запуск исполняемого файла

Далее изменю текст программы и вместо символов, запишу в регистры числа. Исправлю текст программы (Листинг 6.1) (рис. 5.5).

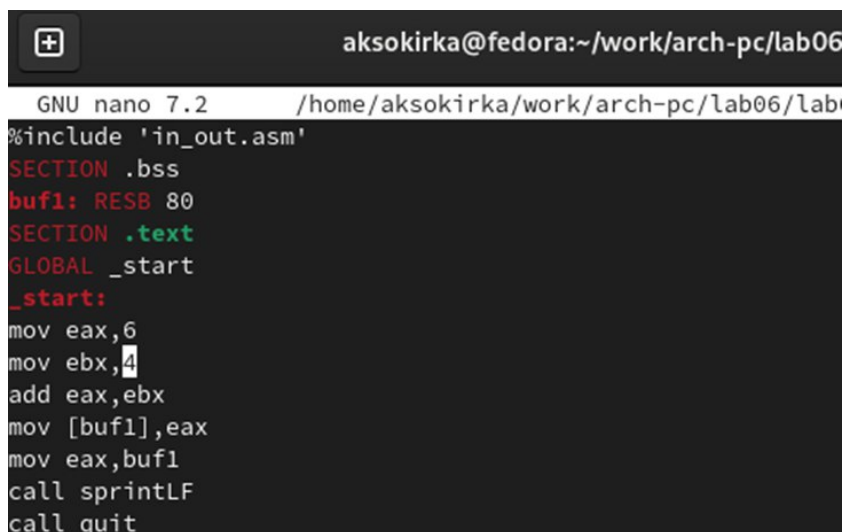


Рис. 5.5: Изменение текста программы

Создам исполняемый файл и запущу его. Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран (рис. 5.6).

```

aksokirka@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
aksokirka@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
aksokirka@fedora:~/work/arch-pc/lab06$ ./lab6-1
aksokirka@fedora:~/work/arch-pc/lab06$

```

Рис. 5.6: Создание файла

Создам файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и введу в него текст программы из листинга 6.2. (рис. 5.7).

```

mc [aksokirka@fedora]:~/work/arch-pc/lab06
GNU nano 7.2 /home/aksokirka/work/arch-pc/lab06/lab6-2.a
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit

```

Рис. 5.7: Создание файла

Создам исполняемый файл и запущу его. 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 (рис. 5.8).

```

aksokirka@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
aksokirka@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
aksokirka@fedora:~/work/arch-pc/lab06$ ./lab6-2
106
aksokirka@fedora:~/work/arch-pc/lab06$

```

Рис. 5.8: Создание исполняемого файла

Аналогично предыдущему примеру изменю символы на числа (рис. 5.9).



```
aksokirka@fedora:~/work/arch-pc/lab06
GNU nano 7.2 /home/aksokirka/work/arch-pc/lab06/lab6-2.
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

Рис. 5.9: Изменение текста программы

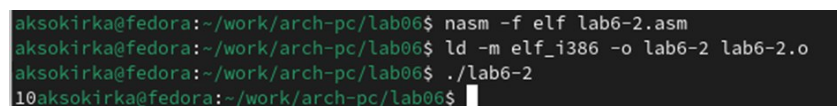
Создам исполняемый файл и запущу его. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10 (рис. 5.10).



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

Рис. 5.10: Создание файла

Заменю функцию iprintLF на iprint. Создам исполняемый файл и запущу его. Вывод изменился. Функция iprint не добавляет к выводу символ переноса строки, в отличие от iprintLF. (рис. 5.11).

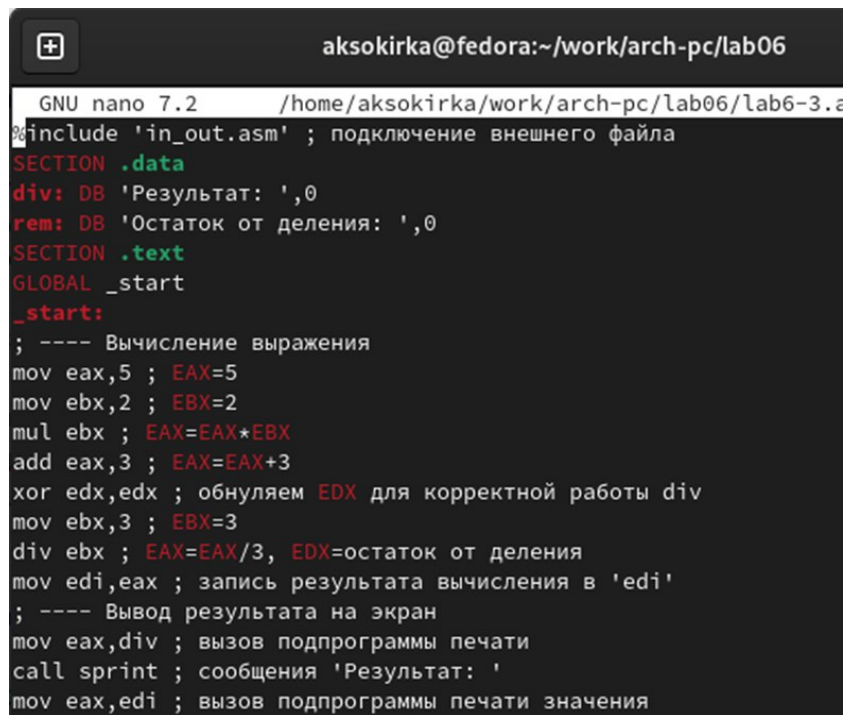


```
aksokirka@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
aksokirka@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
aksokirka@fedora:~/work/arch-pc/lab06$ ./lab6-2
10aksokirka@fedora:~/work/arch-pc/lab06$
```

Рис. 5.11: Изменение текста программы

6 4.2 Выполнение арифметических операций в NASM

Создам файл lab6-3.asm в каталоге ~/work/arch-pc/lab06. Внимательно изучаю текст программы из листинга 6.3 и ввожу в lab6-3.asm (рис. 6.1).



```
aksokirka@fedora:~/work/arch-pc/lab06
GNU nano 7.2 /home/aksokirka/work/arch-pc/lab06/lab6-3.asm
%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
mov ebx,3 ; EBX=3
div ebx ; EAX=EAX/3, EDX=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
; ---- Вывод результата на экран
mov eax,div ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
```

Рис. 6.1: Создание файла

Создам исполняемый файл и запущу его (рис. 6.2).

```

aksokirka@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
aksokirka@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
aksokirka@fedora:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
aksokirka@fedora:~/work/arch-pc/lab06$

```

Рис. 6.2: Создание и запуск файла

Изменяю текст программы для вычисления выражения $\square(\square) = (4 * 6 + 2)/5$. Создам исполняемый файл и проверю его работу (рис. 6.3).

```

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

```

Рис. 6.3: Проверка исполняемого файла

Внимательно изучаю текст программы из листинга 6.4 и ввожу в файл variant.asm. Создам исполняемый файл и запущу его (рис. 6.4).

```

aksokirka@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant.asm variant.asm.o
ld: невозможно найти variant.asm.o: Нет такого файла или каталога
aksokirka@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
aksokirka@fedora:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132246758
Ваш вариант: 19
aksokirka@fedora:~/work/arch-pc/lab06$

```

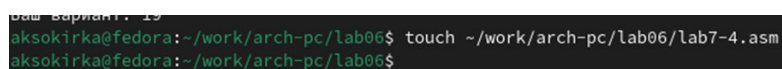
Рис. 6.4: Запуск программы

7 4.2.1 Ответы на вопросы по программе

1. За вывод сообщения “Ваш вариант” отвечают строки кода: `mov eax,rem` `call sprint`
2. Инструкция `mov esx, x` используется, чтобы положить адрес вводимой строки `x` в регистр `esx`. `mov edx, 80` - запись в регистр `edx` длины вводимой строки `call sread` - вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры
3. `call atoi` используется для вызова подпрограммы из внешнего файла, которая преобразует `ascii`-код символа в целое число и записывает результат в регистр `eax`
4. За вычисления варианта отвечают строки: `xor edx,edx` ; обнуление `edx` для корректной работы `div` `mov ebx,20` ; `ebx = 20` `div ebx` ; `eax = eax/20`, `edx` - остаток от деления `inc edx` ; `edx = edx + 1`
5. При выполнении инструкции `div ebx` остаток от деления записывается в регистр `edx`
6. Инструкция `inc edx` увеличивает значение регистра `edx` на 1
7. За вывод на экран результатов вычислений отвечают строки: `mov eax,edx` `call iprintLF`

8 4.3 Выполнение заданий для самостоятельной работы

Создаю файл lab7-4.asm (рис. 8.1).



```
ваш вариант: 19
aksokirka@fedora:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab7-4.asm
aksokirka@fedora:~/work/arch-pc/lab06$
```

Рис. 8.1: Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения $(1/3 \cdot + 5)7$. Это выражение было под вариантом 19 (рис. 8.2).

```
aksokirka@fedora:~/work/arch-pc/lab06
GNU nano 7.2 /home/aksokirka/work/arch-pc/lab06/lab6-4.asm
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0
SECTION .bss
x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры
SECTION .text
GLOBAL _start
_start:
; --- Вычисление выражения
mov eax,msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax,x ;
call atoi ;
xor edx,edx;
mov ebx,3 ;
div ebx ;
add eax,5 ;
mov ebx,7
mul ebx ;
mov edi,eax ; запись результата вычисления в 'edi'
; -- Вывод результата на экран
mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов
call quit ; вызов подпрограммы завершения
```

Рис. 8.2: Ввод текста программы

Создаю и запускаю исполняемый файл (рис. 8.3).

```
aksokirka@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
aksokirka@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
aksokirka@fedora:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 3
Результат: 42
aksokirka@fedora:~/work/arch-pc/lab06$
```

Рис. 8.3: Запуск исполняемого файла

Провожу еще один запуск исполняемого файла для проверки работы программы с другим значением на входе (рис. 8.4).

```
aksokirka@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-4.asm
aksokirka@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-4 lab6-4.o
aksokirka@fedora:~/work/arch-pc/lab06$ ./lab6-4
Введите значение переменной x: 9
Результат: 56
aksokirka@fedora:~/work/arch-pc/lab06$
```

Рис. 8.4: Проверка работы исполняемого файла

9 Выводы

При выполнении данной лабораторной работы я освоила арифметические инструкции языка ассемблера NASM.

10 Список литературы

https://esystem.rudn.ru/pluginfile.php/2089086/mod_resource/content/0/Лабораторная%20работа%20№6.%20Арифметические%20операции%20в%20NASM..pdf