

# **Отчёта по лабораторной работе №6**

**Архитектура компьютера**

Еремина Оксана Андреевна НКАбд-02-23

# Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
	Список литературы	17

## **Список таблиц**

## Список иллюстраций

4.1	Создание папки и файла . . . . .	8
4.2	Создание копии файла . . . . .	8
4.3	Содержимое файла lab6-1.asm . . . . .	9
4.4	Запуск исполняемого файла . . . . .	9
4.5	Редактирование файла lab6-1.asm . . . . .	9
4.6	Запуск исполняемого файла . . . . .	10
4.7	Создание файла . . . . .	10
4.8	Содержимое файла lab6-2.asm . . . . .	10
4.9	Запуск исполняемого файла . . . . .	10
4.10	Редактирование файла lab6-2.asm . . . . .	11
4.11	Запуск исполняемого файла . . . . .	11
4.12	Редактирование файла lab6-2.asm . . . . .	11
4.13	Запуск исполняемого файла . . . . .	11
4.14	Создание файла . . . . .	12
4.15	Содержимое файла lab6-3.asm . . . . .	12
4.16	Запуск исполняемого файла . . . . .	12
4.17	Содержимое файла lab6-3.asm . . . . .	13
4.18	Запуск исполняемого файла . . . . .	13
4.19	Создание файла . . . . .	13
4.20	Редактирование файла variant.asm . . . . .	14
4.21	Запуск исполняемого файла . . . . .	14
4.22	Содержание файла lab6-4.asm . . . . .	16

# 1 Цель работы

Целью данной лабораторной работы является 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. 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.

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.

## 4 Выполнение лабораторной работы

### 1. Символьные и численные данные в NASM

Создаю каталог для программы данной лабораторной работы, и в нем создаю файл lab6-1.asm. И проверяю правильность выполнения команд.(рис.1)

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

Рис. 4.1: Создание папки и файла

Копирую в созданный каталог файл in\_out.asm, т.к. он будет использоваться далее. (рис.2)

```
oaeremina@oaeremina:~/work/arch-pc/lab06$ cp ~/Загрузки/in_out.asm in_out.asm
oaeremina@oaeremina:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1.asm
```

Рис. 4.2: Создание копии файла

Открываю созданный файл, вставляю в него программу вывода значения регистра eax. (рис.3)



```

1 %include 'in_out.asm'
2 SECTION .bss
3 buf1: RESB 80
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov eax, '6'
8 mov ebx, '4'
9 add eax, ebx
10 mov [buf1], eax
11 mov eax, buf1
12 call sprintLF
13 call quit

```

Рис. 4.3: Содержимое файла lab6-1.asm

Создаю исполняемый файл и запускаю его (рис.4)

```

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

```

Рис. 4.4: Запуск исполняемого файла

Изменяю в тексте программы символы '6' и '4' на 6 и 4 (рис.5)

```

1 %include 'in_out.asm'
2 SECTION .bss
3 buf1: RESB 80
4 SECTION .text
5 GLOBAL _start
6 _start:
7 mov eax, 6
8 mov ebx, 4
9 add eax, ebx
10 mov [buf1], eax
11 mov eax, buf1
12 call sprintLF
13 call quit

```

Рис. 4.5: Редактирование файла lab6-1.asm

Создаю исполняемый файл программы и запускаю его. Вывелся символ с кодом 10 -символ перевода строки, он не отображается на экране при выводе (рис.6)

```

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

```

Рис. 4.6: Запуск исполняемого файла

Создаю новый файл с помощью touch (рис.7)

```

oaoeremina@oaoeremina:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-2.asm
oaoeremina@oaoeremina:~/work/arch-pc/lab06$ ls
in_out.asm lab6-1 lab6-1.asm lab6-1.o lab6-2.asm

```

Рис. 4.7: Создание файла

Открываю созданный файл, вставляю в него текст другой программы для вывода значения регистра eax. (рис.8)

```

1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax, '6'
6 mov ebx, '4'
7 add eax, ebx
8 call iprintLF
9 call quit

```

Рис. 4.8: Содержимое файла lab6-2.asm

Создаю и запускаю исполняемый файл lab7-2 (рис.9). Теперь программа выводит число 106, потому что программа позволяет вывести именно число, а не символ, однако происходит именно сложение кодов символов “6” и “4”.

```

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

```

Рис. 4.9: Запуск исполняемого файла

Изменяю в тексте программы символы ‘6’ и ‘4’ на 6 и 4 (рис.10)

```

1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax,6
6 mov ebx,4
7 add eax,ebx
8 call iprintLF
9 call quit

```

Рис. 4.10: Редактирование файла lab6-2.asm

Создаю и запускаю исполняемый файл (рис.11)

```

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

```

Рис. 4.11: Запуск исполняемого файла

Заменяю в тексте программы функцию iprintLF на iprint (рис.12)

```

1 %include 'in_out.asm'
2 SECTION .text
3 GLOBAL _start
4 _start:
5 mov eax,6
6 mov ebx,4
7 add eax,ebx
8 call iprint
9 call quit

```

Рис. 4.12: Редактирование файла lab6-2.asm

Создаю и запускаю новый исполняемый файл (рис.13). Вывод не изменится, потому что символ переноса строки не отображался, когда программа исполнялась с функцией iprintLF, а iprint не добавляет к выводу символ переноса строки, в отличие от iprintLF.

```

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

```

Рис. 4.13: Запуск исполняемого файла

## 2. Выполнение арифметических операций в NASM

Создаю файл lab6-3.asm с помощью touch (рис.14)

```
oaoeremina@oaoeremina:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-3.asm
oaoeremina@oaoeremina:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1.asm  lab6-2     lab6-2.o
lab6-1      lab6-1.o    lab6-2.asm lab6-3.asm
```

Рис. 4.14: Создание файла

Ввожу в созданный файл текст программы для вычисления значения выражения  $f(x) = (5 * 2 + 3)/3$  (рис.15)

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

Рис. 4.15: Содержимое файла lab6-3.asm

Создаю и запускаю новый исполняемый файл (рис.16)

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

Рис. 4.16: Запуск исполняемого файла

Изменяю текст программы для вычисления значения выражения  $f(x) = (4 * 6 + 2)/5$  (рис.17)

```

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

```

Рис. 4.17: Содержимое файла lab6-3.asm

Создаю и запускаю новый исполняемый файл (рис.18)

```

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

```

Рис. 4.18: Запуск исполняемого файла

Создаю файл variant.asm с помощью touch (рис.19)

```

oaeremina@oaeremina:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/variant.asm
oaeremina@oaeremina:~/work/arch-pc/lab06$ ls
in_out.asm  lab6-1.asm  lab6-2  lab6-2.o  lab6-3.asm  variant.asm
lab6-1      lab6-1.o    lab6-2.asm  lab6-3  lab6-3.o

```

Рис. 4.19: Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис.20)

```

1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите № студенческого билета: ',0
4 rem: DB 'Ваш вариант: ',0
5 SECTION .bss
6 x: RESB 80
7 SECTION .text
8 GLOBAL _start
9 _start:
10 mov eax, msg
11 call sprintLF
12 mov ecx, x
13 mov edx, 80
14 call sread
15 mov eax, x ; вызов подпрограммы преобразования
16 call atoi ; ASCII кода в число, 'eax=x'
17 xor edx,edx
18 mov ebx,20
19 div ebx
20 inc edx
21 mov eax,rem
22 call sprint
23 mov eax,edx
24 call iprintLF
25 call quit

```

Рис. 4.20: Редактирование файла variant.asm

Создаю и запускаю исполняемый файл (рис.21). Ввожу номер своего студенческого билета, программа вывела, что мой вариант - 17.

```

oaoeremina@oaoeremina:~/work/arch-pc/lab06$ nasm -f elf variant.asm
oaoeremina@oaoeremina:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
oaoeremina@oaoeremina:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132236056
Ваш вариант: 17

```

Рис. 4.21: Запуск исполняемого файла

Ответы на вопросы:

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

За вывод сообщения "Ваш вариант" отвечают строки кода: `mov eax,rem` `call sprint`

2. Для чего используются следующие инструкции? `mov ecx, x` `mov edx, 80` `call sread`

Инструкция `mov ecx, x` используется, чтобы положить адрес вводимой строки `x` в регистр `ecx` `mov edx, 80` - запись в регистр `edx` длины вводимой строки `call sread`

- вызов подпрограммы из внешнего файла, обеспечивающей ввод сообщения с клавиатуры

3. Для чего используется инструкция “call atoi”?

call atoi используется для вызова подпрограммы из внешнего файла, которая преобразует ascii-код символа в целое число и записывает результат в регистр eax

4. Какие строки листинга 6.4 отвечают за вычисления варианта?

За вычисления варианта отвечают строки: xor edx,edx mov ebx,20 div ebx inc edx

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

При выполнении инструкции div ebx остаток от деления записывается в регистр edx

6. Для чего используется инструкция “inc edx”?

Инструкция inc edx увеличивает значение регистра edx на 1

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

За вывод на экран результатов вычислений отвечают строки: mov eax,edx call iprintLF

3. Выполнение заданий для самостоятельной работы

Создаю файл lab6-4.asm с помощью touch. Открываю созданный файл, ввожу текст программы для вычисления значения выражения  $(11 + x) * 2 - 6$ . Выражение под вариантом 8. (рис. 22)

```

1 %include 'in_out.asm' ; подключение внешнего файла
2 SECTION .data
3 msg: DB 'Введите значение переменной x: ',0
4 rem: DB 'Результат: ',0
5 SECTION .bss
6 x: RESB 80
7 SECTION .text
8 GLOBAL _start
9 _start:
10 ; ---- Вычисление выражения
11 mov eax, msg
12 call sprint
13 mov ecx, x
14 mov edx, 80
15 call sread
16 mov eax, x
17 call atoi
18 add eax, 11
19 mov ebx, 2
20 mul ebx
21 add eax, -6
22 mov edi, eax
23 mov eax, rem ; вызов подпрограммы печати
24 call sprint ; сообщения 'Результат: '
25 mov eax, edi ; вызов подпрограммы печати значения
26 call iprintLF ; из 'edi' в виде символов
27 call quit ; вызов подпрограммы завершения

```

Рис. 4.22: Содержание файла lab6-4.asm

Создаю и запускаю исполняемый файл. При вводе значения 2, программа выводит 20. (рис. 23) # Вывод При выполнении данной лабораторной работы я освоила арифметические инструкции языка ассемблера NASM.



# Список литературы

Архитектура ЭВМ