

Отчет по лабораторной работе №6

Арифметические операции в NASM.

Корчагин Алексей Павлович

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Задания самостоятельной работы	18
4	Выводы	23
	Список литературы	24

Список иллюстраций

2.1	Создание файла и каталога	7
2.2	Копирование файла	7
2.3	Ввод код в файл	8
2.4	Создание и запуск исполняемого файла	8
2.5	Редактирование кода	9
2.6	Запуск изменённого файла	10
2.7	Создание файла	10
2.8	Запуск файла	11
2.9	Редактирование файла	11
2.10	Запуск файла	12
2.11	Редактирование файла	12
2.12	Запуск файла	13
2.13	Создание файла	13
2.14	Редактирование файла	14
2.15	Запуск файла	14
2.16	Редактирование файла	15
2.17	Запуск файла	15
2.18	Создание файла	16
2.19	Редактирование файла	16
3.1	Создание файла	18
3.2	Написание кода для файла	19
3.3	Запуск и тест файла	20

Список таблиц

1 Цель работы

Цель лабораторной работы - освоение арифметических инструкций языка ассемблера NASM.

2 Теоретическое введение

Большая часть 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.

Основные способы адресации в NASM:

Регистровая адресация – операнды хранятся в регистрах и в команде используются имена

Непосредственная адресация – значение операнда задается непосредственно в команде, Н

Адресация памяти – операнд задает адрес в памяти. В команде указывается символическое

Арифметические операции в NASM:

Целочисленное сложение - `add`.

Целочисленное вычитание - `sub`.

Команды инкремент(`inc`)- прибавление единицы к операнду и декремент(`dec`)- вычитание единицы. Они выгодны тем, что они занимают меньше места, чем соответствующи

Команда изменения знака операнда - `neg`.

Команды умножения - `mul` (для беззнакового умножения) и `imul` (для знакового умножения).

Команды деления `div` и `idiv`.

Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. 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 # Выполнение лабораторной работы

Создал каталог для программ лабораторной работы No 6, перешёл в него и создал файл (рис. 2.1).

```
apkorchagin@dk8n56 ~ $ mkdir ~/work/arch-pc/lab06
apkorchagin@dk8n56 ~ $ cd ~/work/arch-pc/lab06
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ touch lab6-1.asm
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $
```

Рис. 2.1: Создание файла и каталога

Скопировал в текущий каталог файл in_out.asm с помощью команды cp, т.к. он будет использоваться в других программах(рис. 2.2).

```
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ cp ~/Заргузки/in_out.asm in_out.asm
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ ls
in_out.asm lab6-1.asm
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $
```

Рис. 2.2: Копирование файла

Открываю файл lab6-1.asm и ввожу в него программу вывода значения регистра eax(рис. 2.3).

```
%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. 2.3: Ввод код в файл

Создайте исполняемый файл и запустите его. Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6 (рис. 2.4).

```
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ ./lab6-1
j
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $
```

Рис. 2.4: Создание и запуск исполняемого файла

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. 2.5).

```
%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. 2.5: Редактирование кода

Создаю новый исполняемый файл программы и запускаю его. Выводится символ с кодом 10, это символ перевода строки. Этот символ не отображается при выводе на экран(рис. 2.6).

```
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ ./lab6-1

apkorchagin@dk8n56 ~/work/arch-pc/lab06 $
```

Рис. 2.6: Запуск изменённого файла

Создаю новый файл и lab6-2.asm и вписываю в него код(рис. 2.7).

```
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. 2.7: Создание файла

Создаю и запускаю исполняемый файл lab6-2. Теперь вывод число 106, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4” (рис. 2.8).

```
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ touch lab6-2.asm
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ ./lab6-2
106
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $
```

Рис. 2.8: Запуск файла

Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4(рис. 2.9).

```
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. 2.9: Редактирование файла

Создаю и запускаю новый исполняемый файл. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10 (рис. 2.10).

```
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ ./lab6-2
10
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $
```

Рис. 2.10: Запуск файла

Заменяю в тексте программы функцию `iprintLF` на `iprint` (рис. 2.11).

```
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. 2.11: Редактирование файла

Создаю и запускаю новый исполняемый файл. Вывод не изменился, 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`. (рис. 2.12).

```
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ ./lab6-2
10apkorchagin@dk8n56 ~/work/arch-pc/lab06 $
```

Рис. 2.12: Запуск файла

Создаю файл lab6-3.asm (рис. 2.13).

```
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ touch lab6-3.asm
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $
```

Рис. 2.13: Создание файла

Ввожу в созданный файл текст программы для вычисления значения выражения $(5 * 2 + 3)/3$ (рис. 2.14).

```
*lab6-3.asm
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
```

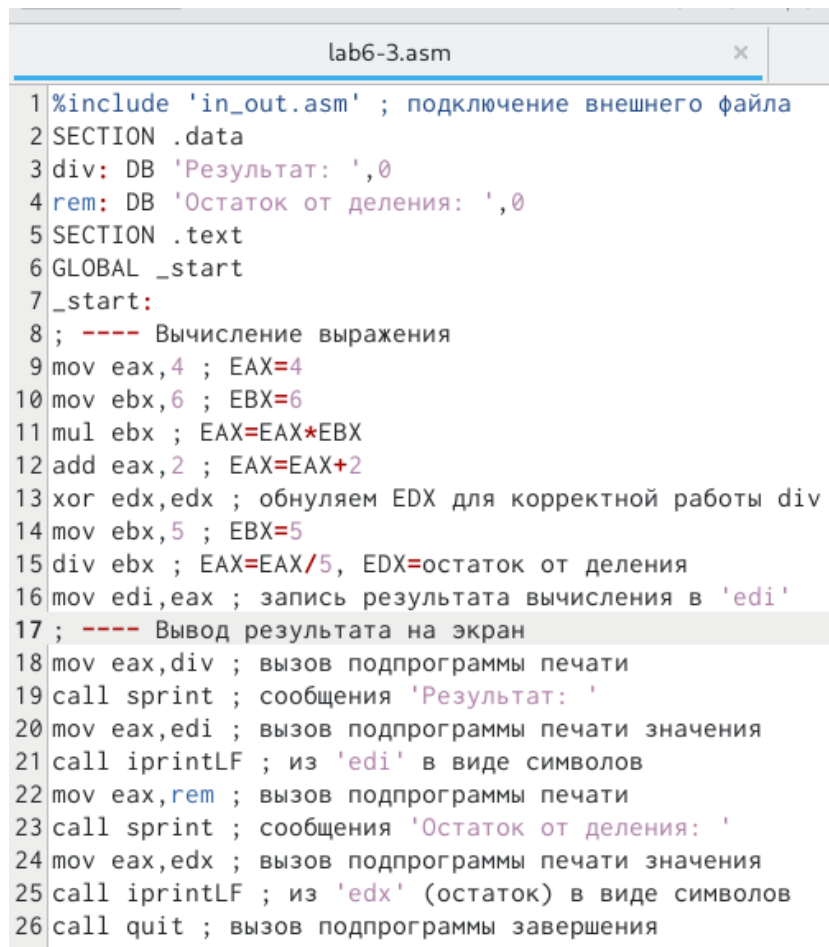
Рис. 2.14: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. 2.15).

```
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ touch lab6-3.asm
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $
```

Рис. 2.15: Запуск файла

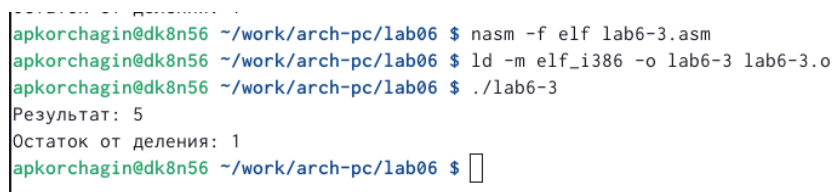
Изменяю программу так, чтобы она вычисляла значение выражения $(4 * 6 + 2)/5$ (рис. 2.16).



```
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
```

Рис. 2.16: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 2.17).



```
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $
```

Рис. 2.17: Запуск файла

Создаю файл variant.asm (рис. 2.18).

```
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ touch variant.asm
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $
```

Рис. 2.18: Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. 2.19).

```
1 %include 'in_out.asm'
2 SECTION .data
3 msg: DB 'Введите No 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 sprintf
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 sprintf
23 mov eax, edx
24 call iprintLF
25 call quit
```

Рис. 2.19: Редактирование файла

Создаю и запускаю исполняемый файл. Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 2(рис. ??).


```

apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ touch variant.asm
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ ./variant
Введите No студенческого билета:
1132236121
Ваш вариант: 2
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ 

```

Ответы на вопросы

1: За вывод сообщения “Ваш вариант” отвечают строки кода: `mov eax,rem` `call sprint`.

2: `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: `call atoi` - вызов 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` прибавляет 1 к значению регистра `edx`.

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

3 Задания самостоятельной работы

Создаю файл lab6-4.asm (рис. 3.1).

```
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $ touch lab6-4.asm
apkorchagin@dk8n56 ~/work/arch-pc/lab06 $
```

Рис. 3.1: Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения $(12x+3)/5$ (рис. 3.2).

```

section .data
    prompt db "Enter a value for x: ", 0
    result_msg db "The result is: ", 0
    remainder_msg db "The remainder is: ", 0
    newline db 10, 0

section .bss
    x resb 1
    result resb 10
    remainder resb 10

section .text
    global _start

_start:
    ; Prompt the user to enter a value for x
    mov eax, 4
    mov ebx, 1
    mov ecx, prompt
    mov edx, 20
    int 0x80

    ; Read the value of x from the user
    mov eax, 3
    mov ebx, 0
    mov ecx, x
    mov edx, 1
    int 0x80

    sub byte [x], '0'

    ; Calculate the value of the expression (12x+3)/5
    mov al, [x]
    mov bl, 12
    mul bl
    add al, 3
    mov bl, 5
    div bl

    add al, '0'
    mov [result], al

    ; Calculate the remainder of the division

```

Рис. 3.2: Написание кода для файла

Создаю и запускаю исполняемый файл. Проверяю с значением из зада-

ния(рис. 3.3).

```
apkorchagin@dk5n52 ~/work/arch-pc/lab06 $ nasm -f elf lab6-4.asm
apkorchagin@dk5n52 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-4 lab6-4.o
apkorchagin@dk5n52 ~/work/arch-pc/lab06 $ ./lab6-4
Enter a value for x:1
The result is:3
The remainder is:0apkorchagin@dk5n52 ~/work/arch-pc/lab06 $
```

Рис. 3.3: Запуск и тест файла

#Листинг программы

1_Старт программы и задание приветственного сообщения

```
section .data prompt db "Enter a value for x:", 0 result_msg db "The result is:", 0
remainder_msg db "The remainder is:", 0 newline db 10, 0
```

2_Задание переменной x

```
section .bss x resb 1 result resb 10 remainder resb 10
section .text global _start
_start: 3_Вывод приветственного сообщения mov eax, 4 mov ebx, 1 mov ecx,
prompt mov edx, 20 int 0x80
```

4_Считывание переменной x

```
mov eax, 3
mov ebx, 0
mov ecx, x
mov edx, 1
int 0x80
```

```
sub byte [x], '0'
```

5_Вычисление значения выражения $(12x+3)/5$ для введённого x

```
mov al, [x]
mov bl, 12
mul bl
add al, 3
mov bl, 5
div bl
```

```
add al, '0'
mov [result], al
```

6_Вычисление остатка от деления

```
mov al, ah
add al, '0'
mov [remainder], al
```

7_Вывод остатка и результата вычисления выражения в консоль

```
mov eax, 4
mov ebx, 1
mov ecx, result_msg
mov edx, 14
int 0x80
```

```
mov eax, 4
mov ebx, 1
mov ecx, result
mov edx, 1
int 0x80
```

```
mov eax, 4
mov ebx, 1
mov ecx, newline
mov edx, 1
int 0x80
```

```
mov eax, 4
mov ebx, 1
mov ecx, remainder_msg
mov edx, 17
int 0x80
```

```
mov eax, 4
mov ebx, 1
mov ecx, remainder
mov edx, 1
int 0x80
```

8_Завершение программы

```
mov eax, 1
xor ebx, ebx
int 0x80
```

4 Выводы

Я научился выполнять базовые арифметические действия при программирование на языке Ассемблера NASM

Список литературы