

Отчет по лабораторной работе №6

Дисциплина: Архитектура компьютера

Никуленков Степан Сергеевич

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Символьные и численные данные в NASM	9
4.2	2 Выполнение арифметических операций в NASM	12
5	Ответы на вопросы по программе	16
6	Выполнение заданий для самостоятельной работы	18
7	Вывод	21
8	Список литературы.	22

Список иллюстраций

4.1	Создание рабочего каталога и файла.	9
4.2	Код программы	9
4.3	Запуск файла	10
4.4	Запуск файла	10
4.5	Запуск файла	10
4.6	Редактирование файла	11
4.7	Запуск файла	11
4.8	Редактирование файла	12
4.9	Запуск файла	12
4.10	Запуск файла	13
4.11	Редактирование файла	14
4.12	Запуск файла	15
4.13	Запуск файла	15
6.1	Запуск файла	19
6.2	Запуск файла	20

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

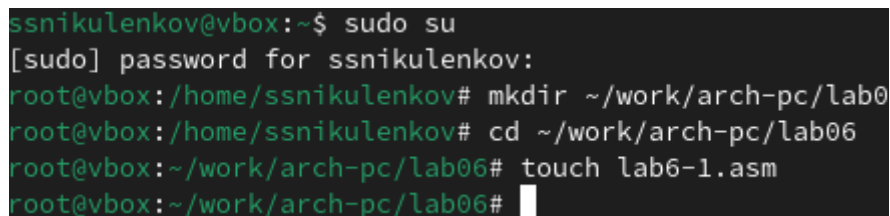
Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда предоставляет место, где хранятся данные, подлежащие обработке. Это могут быть данные хранящиеся в регистре или в ячейке памяти. - Регистровая адресация – операнды хранятся в регистрах и в команде используются имена 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

4.1 Символьные и численные данные в NASM

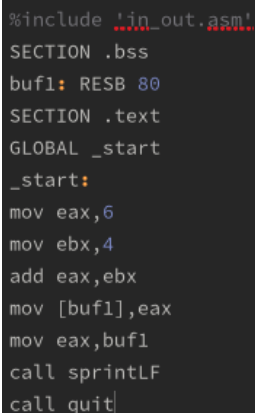
Создаю рабочий каталог `~/work/arch-pc/lab06`, перехожу в него и создаю там файл `lab6-1.asm` (рис. 4.1).



```
ssnikulenkov@vbox:~$ sudo su
[sudo] password for ssnikulenkov:
root@vbox:/home/ssnikulenkov# mkdir ~/work/arch-pc/lab06
root@vbox:/home/ssnikulenkov# cd ~/work/arch-pc/lab06
root@vbox:~/work/arch-pc/lab06# touch lab6-1.asm
root@vbox:~/work/arch-pc/lab06#
```

Рис. 4.1: Создание рабочего каталога и файла.

Перехожу в `lab6-1.asm` и редактирую его (рис. 4.2).



```
%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: Код программы

Создаю исполняемый файл программы и запускаю его (рис. 4.3). Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.

```
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06# nasm -f elf lab6-1.asm
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06# ld -m elf_i386 -o lab6-1 lab6-1.o
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06# ./lab6-1
j
```

Рис. 4.3: Запуск файла

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4. Создаю новый исполняемый файл программы и запускаю его (рис. 4.4). Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

```
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06# nasm -f elf lab6-1.asm
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06# ld -m elf_i386 -o lab6-1 lab6-1.o
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06# ./lab6-1

root@vbox:/home/ssnikulenkov/work/arch-pc/lab06#
```

Рис. 4.4: Запуск файла

Создаю новый файл lab7-2.asm с помощью утилиты touch. Ввожу в файл текст другой программы для вывода значения регистра eax. Создаю и запускаю исполняемый файл lab7-2 (рис. 4.5). 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”.

```
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06# nasm -f elf lab6-2.asm
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06# ld -m elf_i386 -o lab6-2 lab6-2.o
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06# ./lab6-2
106
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06#
```

Рис. 4.5: Запуск файла

Заменяю в тексте программы в файле lab7-2.asm символы “6” и “4” на числа 6 и 4 (рис. 4.6).

```
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprintLF
call quit
```

Рис. 4.6: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 4.7).Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа,поэтому вывод 10.

```
root@vbox:/home/ssnikulenkoy/work/arch-pc/lab06# nasm -f elf lab6-2.asm
root@vbox:/home/ssnikulenkoy/work/arch-pc/lab06# ld -m elf_i386 -o lab6-2 lab6-2.o
root@vbox:/home/ssnikulenkoy/work/arch-pc/lab06# ./lab6-2
10
root@vbox:/home/ssnikulenkoy/work/arch-pc/lab06#
```

Рис. 4.7: Запуск файла

Заменяю в тексте программы функцию iprintLF на iprint (рис. 4.8)

```
%include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,6
mov ebx,4
add eax,ebx
call iprint
call quit
```

Рис. 4.8: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 4.9). 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`.

```
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06# nasm -f elf lab6-2.asm
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06# ld -m elf_i386 -o lab6-2 lab6-2.o
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06# ./lab6-2
10root@vbox:/home/ssnikulenkov/work/arch-pc/lab06#
```

Рис. 4.9: Запуск файла

4.2 2 Выполнение арифметических операций в NASM

Создаю файл `lab7-3.asm` с помощью утилиты `touch`. Ввожу в созданный файл текст программы для вычисления значения выражения $f(x) = (5 * 2 + 3)/3$. Создаю исполняемый файл и запускаю его

```
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06# nasm -f elf lab6-3.asm
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06# ld -m elf_i386 -o lab6-3 lab6-3.o
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06# ./lab6-3
Результат: 4
Остаток от деления: 1
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06#
```

Рис. 4.10: Запуск файла

Изменяю программу так, чтобы она вычисляла значение выражения $f(x) = (4 * 6 + 2)/5$ (рис. 4.11)

```

%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,4 ; EAX=4
mov ebx,6 ; EBX=6
mul ebx ; EAX=EAX*EBX
add eax,2 ; EAX=EAX+2
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,5 ; EBX=5
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
call iprintLF ; из 'edi' в виде символов
mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Остаток от деления: '
mov eax,edx ; вызов подпрограммы печати значения
call iprintLF ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения

```

Рис. 4.11: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. 4.12). Я посчитал для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно.

```

root@vbox:/home/ssnikulenkov/work/arch-pc/lab06# nasm -f elf lab6-3.asm
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06# ld -m elf_i386 -o lab6-3 lab6-3
.o
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06# ./lab6-3
Результат: 5
Остаток от деления: 1
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06#

```

Рис. 4.12: Запуск файла

Создаю файл `variant.asm` с помощью утилиты `touch`. Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета. Создаю и запускаю исполняемый файл. Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 5.

```

root@vbox:/home/ssnikulenkov/work/arch-pc/lab06# nasm -f elf variant.asm
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06# ld -m elf_i386 -o variant variant.o
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06# ./variant
Введите № студенческого билета:
1132246744
Ваш вариант: 5
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06#

```

Рис. 4.13: Запуск файла

5 Ответы на вопросы по программе

1. За вывод сообщения “Ваш вариант” отвечают строки кода:

```
mov eax, rem  
call sprint
```

2. 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. 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
```

6 Выполнение заданий для самостоятельной работы

Создаю файл variant5.Открываю его и ввожу следующий листинг (рис. 6.1).

```
lab5-1.asm | ЛО5никуленков | ЛО4Никуненков

%include 'in_out.asm'

SECTION .data

msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x
call atoi
mov ebx,9
mul ebx
add eax,-8
xor edx,edx
mov ebx,8
div ebx

mov edi,eax

mov eax,rem
call sprint
mov eax,edi
call iprintLF
call quit
```

Рис. 6.1: Запуск файла

Создаю и запускаю исполняемый файл. При вводе значения 8, вывод - 8. При вводе значения 64, вывод - 71.

```
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06# nasm -f elf variant5.asm
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06# ld -m elf_i386 -o variant5 vari
ant5.o
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06# ./variant5
Введите значение переменной x: 8
Результат: 8
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06# ./variant5
Введите значение переменной x: 64
Результат: 71
root@vbox:/home/ssnikulenkov/work/arch-pc/lab06#
```

Рис. 6.2: Запуск файла

7 Вывод

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

8 Список литературы.

1. Лабораторная работа №6
2. Таблица ASCII