

Отчет по лабораторной работе номер 6

Архитектура компьютеров и операционные системы.

Вацаев Виктор Андреевич

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
5	Выводы	15
	Список литературы	16

Список иллюстраций

Список таблиц

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

2 Задание

Выполнить лабораторную работу и получить максимально баллов

3 Теоретическое введение

Адресация в NASM связана с указанием места хранения данных для их обработки в инструкциях. Операнды могут храниться либо в регистрах, либо в ячейках памяти. Способы задания адреса хранения операндов называются методами адресации.

Существует три основных метода адресации:

Регистровая адресация – операнды располагаются в регистрах, и инструкции содержат имена этих регистров. Пример:

```
mov ax, bx
```

Непосредственная адресация – значение операнда указывается прямо в инструкции. Пример:

```
mov ax, 2
```

Адресация памяти – операнд указывает адрес в памяти, где хранятся данные. В инструкции используется символическое имя ячейки памяти, с содержимым которой нужно работать.

Например, если определена переменная:

```
intg DD 3 ; Объявление области памяти размером 4 байта с меткой intg
```

то команда:

```
mov eax, [intg]
```

копирует данные из памяти по адресу intg в регистр eax.

Аналогично, команда:

```
mov [intg], eax
```

записывает данные из регистра eax в память по адресу intg.

Рассмотрим также команду:

```
mov eax, intg
```

В этом случае в регистр `eax` будет записан адрес метки `intg`. Если, например, для `intg` выделена память начиная с адреса `0x600144`, то команда:

```
mov eax, intg
```

эквивалентна:

```
mov eax, 0x600144
```

Это означает, что в регистр `eax` записывается адрес `0x600144`.

4 Выполнение лабораторной работы

1. Создаём каталог для программ лабораторной работы № 6, переходим в

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

него и создаём файл lab6-1.asm

2. Я ввожу в файл lab6-1.asm текст программы из листинга 6.1. В этой программе я записываю символ '6' в регистр eax с помощью команды mov eax, '6', а символ '4' — в регистр ebx с помощью команды mov ebx, '4'. Затем я прибавляю значение из регистра ebx к значению в eax (add eax, ebx), и результат сложения сохраняется в eax. Далее я вывожу результат. Поскольку функция sprintf требует, чтобы в eax был записан адрес, мне нужно использовать дополнительную переменную. Для этого я записываю значение из регистра eax в переменную buf1 (mov [buf1], eax), затем записываю адрес переменной buf1 в eax (mov eax, buf1) и вызываю функцию sprintf.

```

GNU nano 7.2 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 sprintLF
call quit

```

[Прочитано 13 строк]

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^_ Замена ^U Вставить ^J Выводить ^_ К строке

```

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

```

3. Теперь я изменяю

текст программы, чтобы вместо символов в регистры записывались числа. Я исправляю текст программы (Листинг 6.1) следующим образом: заменяю строки

mov eax, '6' mov ebx, '4'

на строки

mov eax, 6 mov ebx, 4

```

GNU nano 7.2 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 sprintLF
call quit

```

[Прочитано 13 строк]

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^_ Замена ^U Вставить ^J Выводить ^_ К строке

```

victor@fedora:~/work/arch-pc/lab06$ nano lab6-1.as
victor@fedora:~/work/arch-pc/lab06$ nano lab6-1.as
victor@fedora:~/work/arch-pc/lab06$ nasm -f elf la
victor@fedora:~/work/arch-pc/lab06$ ld -m elf_i386
victor@fedora:~/work/arch-pc/lab06$ ./lab6-1

```

Затем я создаю исполняемый файл и запускаю его. Как и в предыдущем случае,

при выполнении программы я не получаю число 10. Вместо этого выводится символ с кодом 10, что соответствует символу перевода строки в таблице ASCII. Определение символа для кода 10 по таблице ASCII: Код 10 в таблице ASCII соответствует символу перевода строки (LF — Line Feed). Этот символ не отображается как видимый знак на экране, а используется для перехода на новую строку. 4. Я создаю файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и ввожу в него текст программы из Листинга 6.2, с использованием подпроцедур из файла in_out.asm для преобразования ASCII символов в числа и обратно. Эти функции помогут корректно работать с числами и выводить их

```
GNU nano 7.2 lab6-2.asm
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax, '6'
mov ebx, '4'
add eax, ebx
call iprintLF
call quit
```

[Прочитано 9 строк]

^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^\ Замена ^U Вставить ^J Выровнять ^/_ К строке

в требуемом формате.

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

5. Теперь я заменяю

строки:

mov eax, '6' mov ebx, '4'

на строки:

mov eax, 6 mov ebx, 4

Затем я создаю исполняемый файл и запускаю его. При выполнении программы результатом будет отображение числа 10, поскольку выполняется сложение

```

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

```

чисел. Вывод исполняемого файла: На изображении видно, что программа выводит результат 10. Это, скорее всего, результат вычислений, выполненных в моей программе.

Различие между функциями `iprintlnf` и `iprint`:

`iprintlnf` – выводит значение и автоматически добавляет символ перевода строки (код 10),

`iprint` – выводит только значение без добавления символа перевода строки, поэтому следу

6. Я создаю файл `lab6-3.asm` в каталоге `~/work/arch-pc/lab06` и пишу в него про-

```

GNU nano 7.2 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

```

грамму для вычисления выражения

После этого создаю исполняемый файл, компилирую и запускаю его.

```

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

```

Далее изменяю текст программы для вычисления выражения $f(x) = (4 * 6 + 2) /$

```

GNU nano 7.2 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,4 ; EAX=4
mov ebx,6 ; EBX=2
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/5, 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
[ Записано 26 строк ]
^G Справка ^O Записать ^W Поиск ^K Вырезать ^T Выполнить ^C Позиция
^X Выход ^R ЧитФайл ^N Замена ^U Вставить ^J Выводить ^_ К строке

```

5:

```

victor@fedora:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
victor@fedora:~/work/arch-pc/lab06$ nano lab6-3.asm
victor@fedora:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
victor@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3
victor@fedora:~/work/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
victor@fedora:~/work/arch-pc/lab06$

```

Создаю исполняемый файл и проверяю его работу.

7. Создаю файл variant.asm в каталоге ~/work/arch-pc/lab06.

Вначале вывожу запрос на ввод номера студенческого билета. После этого с помощью функции `sread` программа считывает введенное значение с клавиатуры и сохраняет его в виде строки в переменную.

Затем, чтобы преобразовать введенные символы в число, использую функцию `atoi` из файла `in_out.asm`. Эта функция преобразует строку с номером студенческого билета в целое число и сохраняет его в регистре `EAX`.

Далее вычисляю номер варианта по формуле $(S_n \bmod 20) + 1$, где S_n — это номер студенческого билета. Для получения остатка от деления использую команду `div`, где `EAX` делится на 20, а остаток сохраняется в `EDX`. Увеличиваю остаток на 1, чтобы получить номер варианта.

После этого вывожу на экран сообщение с результатом, используя функции `sprint` для текста и `iprintLF` для вывода самого номера варианта с переводом стро-

```

victor@fedora:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/variant.asm
victor@fedora:~/work/arch-pc/lab06$ nano variant.asm
victor@fedora:~/work/arch-pc/lab06$ nasm -f elf variant.asm
victor@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
victor@fedora:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
10355463
Ваш вариант: 4
victor@fedora:~/work/arch-pc/lab06$

```

ки. 8. Самостоятельная

работа! Я напишу программу для вычисления выражения $y=f(x)$. Программа будет выводить выражение для вычисления, запрашивать ввод значения x , вычислять заданное выражение в зависимости от введенного x , а затем выводить результат вычислений. Я выберу вид функции $f(x)$ из таблицы 6.3, в зависимости от номера, который я получу при выполнении лабораторной работы. Я создам исполняемый файл и проверю его работу для значений x_1 и x_2 из 6.3.

```

victor@fedora:~/work/arch-pc/lab06$ nano rabota.asm
victor@fedora:~/work/arch-pc/lab06$ ./rabota
Введите 34
Резул9
victor@fedora:~/work/arch-pc/lab06$ ./rabota
Введите 32
Резул33
!victor@fedora:~/work/arch-pc/lab06$ nano rabota.asm
victor@fedora:~/work/arch-pc/lab06$ ./rabota
Введите 32
Резул33

```

5 Выводы

Заключение:

В ходе выполнения лабораторной работы были рассмотрены различные аспекты программирования на ассемблере, включая работу с регистрами, арифметические операции, вывод данных и использование библиотечных функций. Все задачи были успешно выполнены, что позволяет углубленно понять основы работы с ассемблером, а также взаимодействие с системными функциями для выполнения базовых операций.

Список литературы

None