Отчет по лабораторной работе №6

Арифметические операции в NASM

Ашуров Захид Фамил оглы

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Освоить арифметические инструкции языка ассемблера NASM

2 Задание

Символьные и численные данные в NASM

Выполнение арифметических операций в NASM

3 Теоретическое введение

Адресация в NASM

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес опе

ранда 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 ко- манде, Например: mov ax,2. • Адресация памяти – операнд задает адрес в памяти. В команде указывается символи- ческое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию.

Например, определим переменную intg DD 3 - это означает, что задается область пам

размером 4 байта, адрес которой обозначен меткой intg. В таком случае, команда

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

копирует из памяти по адресу intg данные в регистр eax. В свою очередь команда mov [intg],eax

запишет в память по адресу intg данные из регистра eax.

Также рассмотрим команду

```
mov eax,intg
```

В этом случае в регистр eax запишется адрес intg. Допустим, для intg выделена память начиная с ячейки с адресом 0x600144, тогда команда mov eax,intg аналогична команде mov eax,0x600144 – т.е. эта команда запишет в регистр eax число 0x600144.

Целочисленное сложение add.

```
Схема команды целочисленного сложения add (от англ. addition - добавление) выполн
```

сложение двух операндов и записывает результат по адресу первого операнда. Команда add работает как с числами со знаком, так и без знака и выглядит следующим образом:

```
Допустимые сочетания операндов для команды add аналогичны сочетаниям операндов
```

для команды mov. Так, например, команда add eax,ebx прибавит значение из регистра eax к значению из регистра ebx и запишет результат в регистр eax.

Примеры: add ax,5 ; AX = AX + 5 add dx,cx ; DX = DX + CX add dx,cl ; Ошибка: разный размер операндов.

4 Выполнение лабораторной работы

Создаем каталог для программ лабораторной работы №6, переходим в него и создаем файл lab6-1.asm (Рис. 4.1).

```
zfashurov@dk4n62 ~ $ mkdir ~/work/arch-pc/lab06
zfashurov@dk4n62 ~ $ cd ~/work/arch-pc/lab06
zfashurov@dk4n62 ~/work/arch-pc/lab06 $ touch lab6-1.asm
```

Рис. 4.1: Создание каталога для программ для лабораторной работы №6 и переходим в него и создаем файл lab6-1.asm

Вписываем в lab6-1.asm текст из листинга 6.1 (Рис. 4.2).

Открыть | lab6-1.asm ~/work/arch-pc/lab06 | Сохранить

```
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. 4.2: Вписываем в lab6-1.asm

Создаем файл lab6-1.asm (Рис. 4.3).

```
zfashurov@dk8n54 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
```

Рис. 4.3: Создание файла

Запускаем файл lab6-1.asm (Рис. 4.4).

```
zfashurov@dk8n54 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
zfashurov@dk8n54 ~/work/arch-pc/lab06 $ ./lab6-1
j
```

Рис. 4.4: Запуска файла

Редактируем текст файла lab6-1.asm (Рис. 4.5).

```
%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. 4.5: Редактирование текста файла lab6-1.asm

Создаем и запускаем измененный файл lab6-1.asm (Рис. 4.6).

```
zfashurov@dk8n54 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
zfashurov@dk8n54 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
zfashurov@dk8n54 ~/work/arch-pc/lab06 $ ./lab6-1
```

Рис. 4.6: Создание и запускание измененного файла lab6-1.asm

Создаем файл lab6-2.asm (Рис. 4.7).

```
zfashurov@dk8n54 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-2.asm
```

Рис. 4.7: Создание файла lab6-2.asm

Вписываем текст в lab6-2.asm (Рис. 4.8).

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

Рис. 4.8: Вписывание в текст lab6-2.asm

Создаем и запускаем файл lab6-2.asm (Рис. 4.9).

```
zfashurov@dk8n54 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
zfashurov@dk8n54 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
zfashurov@dk8n54 ~/work/arch-pc/lab06 $ ./lab6-2
106
```

Рис. 4.9: Создание и запускание файла lab6-2.asm

Редактируем lab6-2.asm (Рис. 4.10).

Открыть lab6-2.asm ~/work/arch-pc/lab06 Сохранить

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

Рис. 4.10: Редактирование lab6-2.asm

Создаем и запускаем измененный файл lab6-2.asm (Рис. 4.11).

```
zfashurov@dk8n54 ~/work/arch-pc/lab06 $ gedit lab6-2.asm
zfashurov@dk8n54 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
zfashurov@dk8n54 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
zfashurov@dk8n54 ~/work/arch-pc/lab06 $ ./lab6-2
10
```

Рис. 4.11: Создание и запускание измененного файла lab6-2.asm

Меняем в тексте с inprintLF на inprint (Рис. 4.12).

Открыть lab6-2.asm ~/work/arch-pc/lab06 Сохранить

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

Рис. 4.12: Смена с inprintLF на inprint

Создаем и запускаем файл lab6-2.asm (Рис. 4.13).

```
zfashurov@dk8n54 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
zfashurov@dk8n54 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
zfashurov@dk8n54 ~/work/arch-pc/lab06 $ ./lab6-2
10zfashurov@dk8n54 ~/work/arch-pc/lab06 $
```

Рис. 4.13: Создание и запускание файл lab6-2.asm

- Разница заключается в том что 10 в первом случае была записана отдельно от строки, а вторая слипнулась с 10.

Создаем файл lab6-3.asm (Рис. 4.14).

```
zfashurov@dk8n54 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/lab6-3.asm
```

Рис. 4.14: Создание файла lab6-3.asm

Вписываем в lab6-3.asm (Рис. 4.15).

Открыть | *lab6-3.asm ~/work/arch-pc/lab06 | Сохранить

```
;-------------------------------
; Программа вычисления выражения
;-------------------------------
%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
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. 4.15: Вписывание в lab6-3.asm

Создаем и запускаем файл lab6-3.asm (Рис. 4.16).

```
zfashurov@dk8n54 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
zfashurov@dk8n54 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
zfashurov@dk8n54 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
```

Рис. 4.16: Создание и запускание lab6-3.asm

Редактируем файл lab6-3.asm (Рис. 4.17).

Открыть | lab6-3.asm ~/work/arch-pc/lab06 | Сохранить

```
;-------------------------------
; Программа вычисления выражения
;-------------------------------
%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/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
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. 4.17: Редактирование файла lab6-3.asm

Создаем и запускаем файл lab6-3.asm(Рис. 4.18).

```
zfashurov@dk8n54 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
zfashurov@dk8n54 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
zfashurov@dk8n54 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
```

Рис. 4.18: Создание и запускание файла lab6-3.asm

Создаем файл variant.asm(Рис. 4.19).

```
zfashurov@dk8n54 ~/work/arch-pc/lab06 $ touch ~/work/arch-pc/lab06/variant.asm
```

Рис. 4.19: Создание файла variant.asm

5 Ответы на вопросы по программе

Вопрос 1: За вывод сообщения “Ваш вариант” отвечает строка кода: mov eax,rem call sprint

Вопрос 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.

Вопрос 3: call atoi исользуется для вызова подпрограммы из внешнего файла, которая преобразует ascii-код символа в целое число и записывает результат в регистр eax.

Вопрос 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
```

Вопрос 5: При выполнении инструкции div ebx остаток от деления записывается в регистр edx.

Вопрос 6: Инструкция inc edx увеличивает значение регистра edx на 1

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

6 Выводы

Я освоил арифметические инструкции языка ассемблера NASM

Список литературы