Лабораторная работа №6.

Арифметические операции в NASM.

Самарханова Полина Тимуровна

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

2 Задание

Символьные и численные данные в NASM Выполнение арифметических операций в NASM Задания для самостоятельной работы

3 Теоретическое введение

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.

4 Выполнение лабораторной работы

4.1 Символьные и численные данные в NASM

С помощью команды mkdir я создала новую директорию,в которой далее создавала файлы с программами во время всей лабораторной работы №6 (рис. 4.1).

```
[spolina@fedora ~]$ mkdir ~/work/arch-pc/lab06
[spolina@fedora ~]$ cd  ~/work/arch-pc/lab06
```

Рис. 4.1: Создание каталога

Далее я создала файл lab6-1.asm,используя команду”touch” (рис. 4.2).

```
[spolina@fedora lab06]$ touch lab6-1.asm
[spolina@fedora lab06]$ ls
lab6-1.asm
```

Рис. 4.2: Создание файла

Скопировала в созданный каталог файл in_out.asm,т.к он использовался и в других программах (рис. 4.3).

```
[spolina@fedora lab06]$ cp ~/Загрузки/in_out.asm in_out.asm
[spolina@fedora lab06]$ ls
in_out.asm  lab6-1.asm
```

Рис. 4.3: Копирование файла

После я открыла файл lab6-1.asm в nano и вставила в него программу ввода значения регистра eax (рис. 4.4).

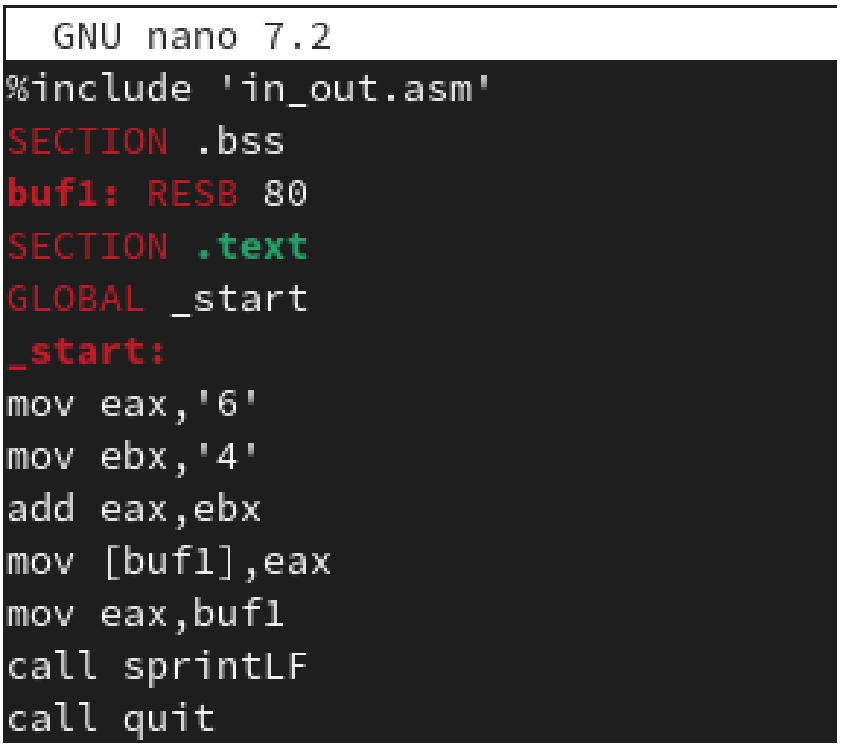

Рис. 4.4: Редактирование файла

Далее я создала объектный файл и после его компановки запустила программу (рис. 4.5). Программа вывела символ “j”, т.к программа вывела символ, который соответствует в системе ASCII сумме двоичных символов 4 и 6.

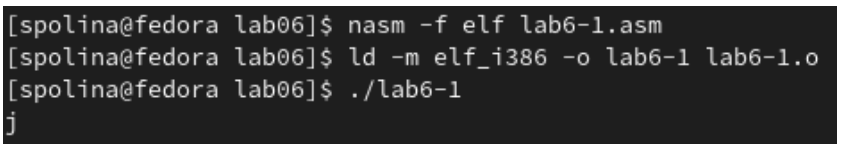

Рис. 4.5: Запуск программы

После я изменила символы ‘6’ и ‘4’ на цифры 6 и 4 (рис. **??**).

```
 GNU nano 7.2
%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 sprintLF
call quit
```

‘ Далее я создала новый исполняемый файл и запустила программу (рис. 4.6). Программа вывела символ с кодом 10,это символ перевода строки, он не отображается при выводе на экран.

```
[spolina@fedora lab06]$ nasm -f elf lab6-1.asm
[spolina@fedora lab06]$ ld -m elf_i386 -o lab6-1 lab6-1.o
[spolina@fedora lab06]$ ./lab6-1
```

Рис. 4.6: Запуск исполняемого файла

После я создала новый файл под названием lab6-2.asm и проверила его наличие(рис. 4.7).

```
[spolina@fedora lab06]$ touch lab6-2.asm
[spolina@fedora lab06]$ ls
in_out.asm  lab6-1  lab6-1.asm  lab6-1.o  lab6-2.asm
```

Рис. 4.7: Создание нового файла

После я ввела в файл текст уже другой программы для вывода eax (рис. 4.8).

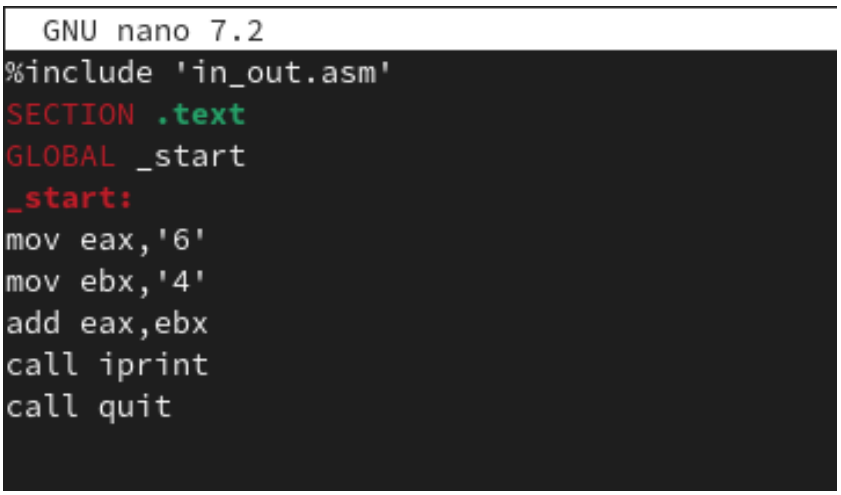

Рис. 4.8: Редактирование программы

Я создала и запустила исполняемый файл lab6-2 (рис. 4.9). Программа стала выводить 106, т.к программа выводит именно число,не символ, хотя всё еще происходит сложение кодов символов “6” и “4”.

Рис. 4.9: Запуск программы

Я заменила в тексте данной программы символы ‘4’ и ‘6’ на числа 6 и 4 (рис. 4.10).

Рис. 4.10: Редактирование программы

Снова создала исполняемый файл программы lab6-2 (рис. 4.11). Теперь программа складывала именно числа,поэтому выводом является сумма 4+6,которая равна 10.

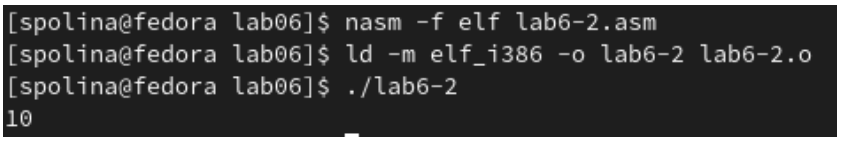

Рис. 4.11: Запуск программы

После я заменила в тексте программы функцию iprintLF на iprint (рис. 4.12).

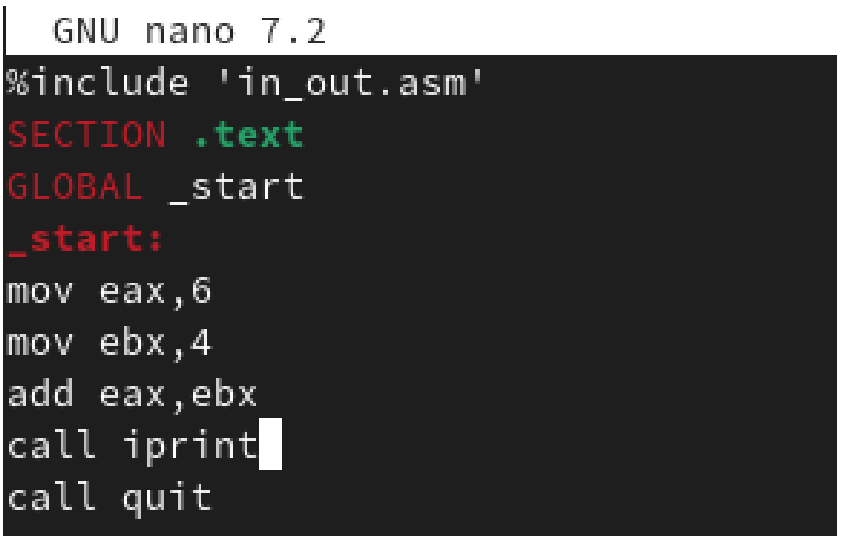

Рис. 4.12: Редактирование программы

Создала и запустила исполняемый файл (рис. 4.13).

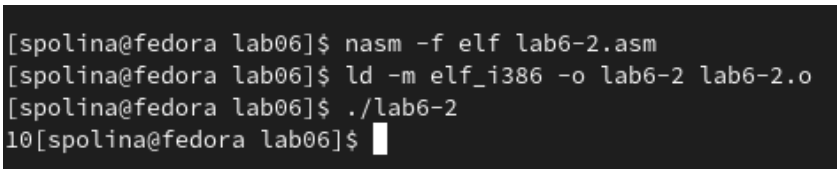

Рис. 4.13: Запуск программы

4.2 Выполнение арифметических операций в NASM

Я создала файл,назвала его lab6-3.asm, используя команду touch (рис. 4.14).

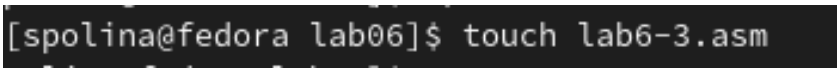

Рис. 4.14: Создание файла

Далее я ввела в созданный файл текст программы для вычисления значения выражения f(x)=(5*2+3)/3 (рис. 4.15).

```
 GNU nano 7.2
%include 'in_out.asm'
SECTION .data
div: DB 'Результат: ',0
rem: DB 'Остаток от деления: ',0
SECTION .text
GLOBAL _start
_start:
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
```

Рис. 4.15: Редактирование программы

Запустила созданный исполняемый файл (рис. 4.16).

```
[spolina@fedora lab06]$ nasm -f elf lab6-3.asm
[spolina@fedora lab06]$ ld -m elf_i386 -o lab6-3 lab6-3.o
[spolina@fedora lab06]$ ./lab6-3
Результат: 4
Остаток от деления: 1
```

Рис. 4.16: Запуск программы

Изменила программу так, чтобы она вычисляла другое выражение f(x)=(4*6+2)/5 (рис. 4.17).

```
 GNU nano 7.2
%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
```

Рис. 4.17: Редактирование программы

Далее я создала исполняемый файл и запустила программу (рис. 4.18). Программа посчитала выражение правильно.

```
[spolina@fedora lab06]$ nasm -f elf lab6-3.asm
[spolina@fedora lab06]$ ld -m elf_i386 -o lab6-3 lab6-3.o
[spolina@fedora lab06]$ ./lab6-3
Результат: 5
Остаток от деления: 1
```

Рис. 4.18: Запуск программы

После создала новый файл variant.asm (рис. 4.19).

```
[spolina@fedora lab06]$ touch variant.asm
```

Рис. 4.19: Создание файла

Ввела в файл текст программы, который вычисляет задания по номеру студенческого билета (рис. 4.20).

```
 GNU nano 7.2
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите № студенческого билета: ',0
rem: DB 'Ваш вариант: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintLF
mov ecx, x
mov edx, 80
call sread
mov eax,x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`
xor edx,edx
mov ebx,20
div ebx
inc edx
mov eax,rem
call sprint
mov eax,edx
call iprintLF
call quit
```

Рис. 4.20: Редактирование программы

Создала исполняемый файл и запустила программу (рис. 4.21). Ввела номер своего студенческого

```
[spolina@fedora lab06]$ nasm -f elf variant.asm
[spolina@fedora lab06]$ ld -m elf_i386 -o variant variant.o
[spolina@fedora lab06]$ ./variant
Введите № студенческого билета:
1132236028
Ваш вариант: 9
```

Рис. 4.21: Запуск программы

4.2.1 Ответы на вопросы

1. За вывод сообщения “Ваш вариант” отвечают строчки кода

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

2. Инструкция mov ec, 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 eax 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
```

4.3 Задания для самостоятельной работы

Я создала файл lab6-4.asm,используя команду touch (рис. 4.22).

```
[spolina@fedora lab06]$ touch lab6-4.asm
```

Рис. 4.22: Создание файла

Открыла файл, ввела в него выражения №9 (рис. 4.23).

```
 GNU nano 7.2
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите значение x: ',0
rem: DB 'Результат: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintLF
mov ecx, x
mov edx, 80
call sread
mov eax,x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`
mov  ebx,31 ; EBX=31
mul ebx;EAX=EAX*EBX=x*31
add eax,-5 ; eax=eax-5=x*31-5
xor edx,edx ; обнуляем EDX для корректной работы div
add eax,10 ;  EAX=EAX+10
mov eax

mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintLF ; из 'edi' в виде символов

call quit ; вызов подпрограммы завершения
```

Рис. 4.23: Редактирование программы

После создала исполняемый файл и запустила программу (рис. 4.24).

Программа вывела правильный ответ.

```
[spolina@fedora lab06]$ nasm -f elf lab6-4.asm
[spolina@fedora lab06]$ ld -m elf_i386 -o lab6-4 lab6-4.o
[spolina@fedora lab06]$ ./lab6-4
Введите значение x:
2
Результат: 67
```

Рис. 4.24: Запуск программы

5 Выводы

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

Список литературы

1. Лабораторная работа №6
2. Таблица ASCII