

Лабораторная работа №7

Арифметические операции в NASM.

Татьяна Алексеевна Коннова, НПИБд-01-22

Содержание

1	Цель работы	4
1.1	Выполнение лабораторной работы	4
2	7.3.2. Выполнение арифметических операций в NASM	14
3	Самостоятельная работа	17
4	Выводы	20

Список иллюстраций

1.1	touch	4
1.2	lab7-1	6
1.3	Output j	7
1.4	Output	8
1.5	Output_prog	9
1.6	Output	9
1.7	Output	9
1.8	LAB7-2	11
1.9	Output	12
1.10	Iprint	13
2.1	Experession	15
3.1	Variant	18
3.2	Zadanie	19

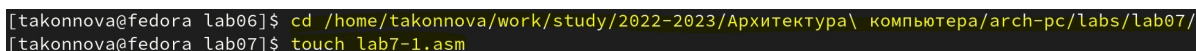
1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM. ## Задание
Выполнить операции с помощью ассемблера

1.1 Выполнение лабораторной работы

1. Создаем каталог для программ лабораторной работы No 7, перейдем в него и создаем файл lab7-1.asm: `mkdir ~/work/arch-pc/lab07 cd ~/work/arch-pc/lab07 touch lab7-1.asm`

(рис. 1.1)



```
[takonnova@fedora lab06]$ cd /home/takonnova/work/study/2022-2023/Архитектура\ компьютера/arch-pc/labs/lab07/
[takonnova@fedora lab07]$ touch lab7-1.asm
```

Рис. 1.1: touch

2. Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения записанные в регистр еах. Введите в файл lab7-1.asm текст программы из листинга 7.1. В данной программе в регистр еах записывается символ 6 (`mov еах, '6'`), в регистр ебх символ 4 (`mov ебх, '4'`). Далее к значению в регистре еах прибавляем значение регистра ебх (`add еах, ебх`, результат сложения запишется в регистр еах). Далее выводим результат. Так как для работы функции `sprintf` в регистр еах должен быть записан адрес, необходимо использовать дополнительную переменную. Для этого запишем значение регистра еах в переменную `buf1`

(mov [buf1],eax), а затем запишем адрес переменной buf1 в регистр eax (mov eax,buf1) и вызовем функцию sprintf

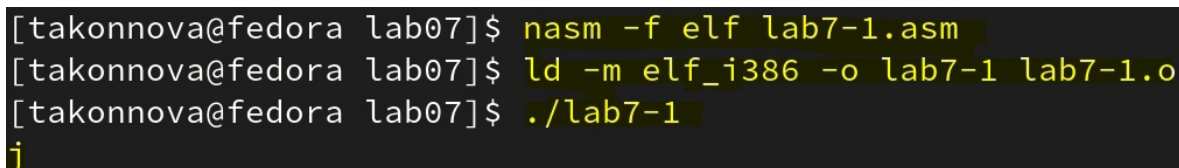
(рис. 1.2)

```
lab7-1.asm [-M
#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. 1.2: lab7-1

В данном случае при выводе значения регистра `eax` мы ожидаем увидеть число 10. Однако результатом будет символ `j`. Это происходит потому, что код символа 6 равен 00110110 в двоичном представлении (или 54 в десятичном представлении), а код символа 4 – 00110100 (52). Команда `add eax,ebx` запишет в регистр `eax` сумму кодов – 01101010 (106), что в свою очередь является кодом символа `j`

(рис. 1.3)



```
[takonnova@fedora lab07]$ nasm -f elf lab7-1.asm
[takonnova@fedora lab07]$ ld -m elf_i386 -o lab7-1 lab7-1.o
[takonnova@fedora lab07]$ ./lab7-1
j
```

Рис. 1.3: Output j

3. Далее изменим текст программы и вместо символов, запишем в регистры числа. Исправим текст программы (Листинг 1) следующим образом:

`mov eax,'6' mov ebx,'4'` на строки `mov eax,6 mov ebx,4`

Как и в предыдущем случае при исполнении программы мы не получим число (рис. 1.4)

```
lab7-1.asm [-M-  
%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. 1.4: Output

(рис. 1.5)

```
[takonnova@fedora lab07]$ nasm -f elf lab7-3.asm
[takonnova@fedora lab07]$ ld -m elf_i386 -o lab7-3 lab7-3.o
[takonnova@fedora lab07]$ ./lab7-3
Результат: 4
Остаток от деления: 1
[takonnova@fedora lab07]$
```

Рис. 1.5: Output_prog

10. В данном случае выводится символ с кодом 10. Пользуясь таблицей ASCII определим какому символу соответствует код 10.

line feed, LF — «подача бумаги на строку») — управляющий символ ASCII (0x0A, 10 в десятичной системе счисления, при выводе которого курсор перемещается на следующую строку.

(рис. 1.6)

```
[takonnova@fedora lab07]$ nasm -f elf lab7-2.asm
[takonnova@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[takonnova@fedora lab07]$ ./lab7-2
10[takonnova@fedora lab07]$
```

Рис. 1.6: Output

(рис. 1.7)

```
[takonnova@fedora lab07]$ nasm -f elf lab7-2.asm
[takonnova@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o
[takonnova@fedora lab07]$ ./lab7-2
10
```

Рис. 1.7: Output

4. Как отмечалось выше, для работы с числами в файле in_out.asm реализованы подпрограммы для преобразования ASCII символов в числа и обратно.

Преобразуем текст программы из Листинга 7.1 с использованием этих функций. Создаем файл lab7-2.asm в каталоге ~/work/arch-pc/lab07 и введем в него текст программы из листинга 7.2. touch ~/work/arch-pc/lab07/lab7-2.asm

Создаем исполняемый файл и запускаем его В результате работы программы мы получим число 106. В данном случае, как и в первом, команда add складывает коды символов '6' и '4' ($54+52=106$). Однако, в отличии от программы из листинга 7.1, функция iprintLF позволяет вывести число, а не символ, кодом которого является это число.

(рис. 1.8)

```
lab7-2.asm [-
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov  eax, '6'
mov  ebx, '4'
add  eax, ebx
call iprintLF
call quit
```

Рис. 1.8: LAB7-2

(рис. 1.9)

```
[takonnova@fedora lab07]$ mc  
  
[takonnova@fedora lab07]$ nasm -f elf lab7-2.asm  
[takonnova@fedora lab07]$ ld -m elf_i386 -o lab7-2 lab7-2.o  
[takonnova@fedora lab07]$ ./lab7-2  
106
```

Рис. 1.9: Output

5. Аналогично предыдущему примеру изменим символы на числа. Заменяем строки

`mov eax,'6' mov ebx,'4'` на строки `mov eax,6 mov ebx,4`

Создаем исполняемый файл и запускаем его. Заменяем функцию `iprintLF` на `iprint`. Создаем исполняемый файл и запускаем его.

(рис. 1.10)

```
lab7-2.asm [-l  
%include 'in_out.asm'  
SECTION .text  
GLOBAL _start  
_start:  
mov eax,6  
mov ebx,4  
add eax,ebx  
call iprint  
call quit
```

Рис. 1.10: Iprint

2 7.3.2. Выполнение арифметических операций в NASM

6. В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения $f(x) = (5 * 2 + 3)/3$. Создаем файл lab7-3.asm в каталоге ~/work/arch-pc/lab07: touch ~/work/arch-pc/lab07/lab7-3.asm Внимательно изучаем текст программы из листинга 7.3 и вводим в lab7-3.asm.

(рис. 2.1)

```

lab7-3.asm          [-M--] 21 L:[ 1+ 0 1/ 29] *(21
;-----
; 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: '

```

Рис. 2.1: Experession

1. Какие строки листинга 7.4 отвечают за вывод на экран сообщения 'Ваш вариант:'?

```

mov eax, rem
call sprint

```

2. Для чего используются следующие инструкции? `mov ecx, x`, `mov edx, 80`, `call sread` Подобное необходимо затем, чтобы мы ввели послы не более 80 символьных значений в переменную `x`

3. Для чего используется инструкция “call atoi”? Чтобы работать не с символами из ASCII работающей по порядковым номерам, а с числами, то есть с реальными цифрами.

4. Какие строки листинга 7.4 отвечают за вычисления варианта?

```
xor edx,edx
```

```
mov ebx,20
```

```
div ebx
```

```
inc edx
```

5. В какой регистр записывается остаток от деления при выполнении инструкции “div ebx”?

```
edx
```

6. Для чего используется инструкция “inc edx”? Чтобы провести требуемую операцию по нахождению остатка от 20 и прибавления единицы. В частности, последнего.

7. Какие строки листинга 7.4 отвечают за вывод на экран результата вычислений?

```
mov eax,edx call iprintLF
```


3 Самостоятельная работа

Выполним самостоятельное задание, следуя инструкции. Произведем вычисления 19 функции, воспользовавшись ассемблером.

(рис. 3.1)

```
variant.asm          [-M-0] 25 L:[ 1+ 1 2/ 28]
;-----
; Программа вычисления варианта
;-----
%include 'in_out.asm'
SECTION .data
msg: DB 'Введите No студенческого билета: ',0
rem: DB 'Ваш вариант: ',0
SECTION .bss
x: RESB 80
SECTION .text
GLOBAL _start
_start:
mov eax, msg
call sprintf
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
```

1Помощь2Сохранить3Блок4Замена5К

Рис. 3.1: Variant

(рис. 3.2)

```
[takonnova@fedora lab07]$ ./variant
Введите No студенческого билета:
1132221818
Ваш вариант: 19
[takonnova@fedora lab07]$ ./samost
(1/3*x+5)*7:
Give me x, please: 3
Rezult is: 42
[takonnova@fedora lab07]$ ./samost
(1/3*x+5)*7:
Give me x, please: 9
Rezult is: 56
[takonnova@fedora lab07]$
```

Рис. 3.2: Zadanie

4 Выводы

Освоение арифметических инструкций языка ассемблера NASM.