

Лабораторная работа №6

Арифметические операции в NASM

Перфилов Александр Константинович | группа: НПИбд 02-23

Содержание

1	Цель работы	5
2	Выполнение Лабораторной работы	6
3	Самостоятельная работа	20
4	Выводы	23

Список иллюстраций

2.1	Рис 2.1.1: Создание каталога и файла .asm	6
2.2	Рис 2.1.2: Демонстрация текста программы в файле	7
2.3	Рис 2.1.3: Создание исполняемого файла и проверка работы . . .	7
2.4	Рис 2.1.4: Измененная программа	8
2.5	Рис 2.1.5: Создание исполняемого файла и проверка работы . . .	9
2.6	Рис 2.1.6: Создание файла	9
2.7	Рис 2.1.7: Программа	10
2.8	Рис 2.1.8: Создание исполняемого файла и проверка работы . . .	10
2.9	Рис 2.1.9: Демонстрация измененной программы	11
2.10	Рис 2.1.10: Создание исполняемого файла и проверка работы . . .	11
2.11	Рис 2.1.11: Именованная программа	12
2.12	Рис 2.1.12: Создание исполняемого файла и проверка работы . . .	12
2.13	Рис 2.2.1: Создание файла	13
2.14	Рис 2.2.2: Программа	13
2.15	Рис 2.2.3: Создание исполняемого файла и проверка работы . . .	14
2.16	Рис 2.2.4: Программа	15
2.17	Рис 2.2.5: Создание исполняемого файла и проверка работы . . .	16
2.18	Рис 2.2.6: Создание файла	16
2.19	Рис 2.2.7: Программа	17
2.20	Рис 2.2.8: Создание исполняемого файла и проверка работы . . .	18
3.1	Рис 3.1.1: Создание файла	20
3.2	Рис 3.1.2: Программа	21
3.3	Рис 3.1.3: Проверка программы	22

Список таблиц

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

2 Выполнение Лабораторной работы

Символьные и численные данные в NASM

Создадим каталог для программ лабораторной работы № 6, перейдем в него и создадим файл lab6-1.asm:

```
perfilov@akperfilov:~/work/study/2023-2024/Архитектура компьютера/arch-pc/labs$ cd
perfilov@akperfilov:~$ mkdir ~/work/arch-pc/lab06
perfilov@akperfilov:~$ cd ~/work/arch-pc/lab06
perfilov@akperfilov:~/work/arch-pc/lab06$ touch lab6-1.asm
perfilov@akperfilov:~/work/arch-pc/lab06$
```

Рис. 2.1: Рис 2.1.1: Создание каталога и файла .asm

Рассмотрим примеры программ вывода символьных и численных значений. Программы будут выводить значения записанные в регистр еах.

Введем в файле lab6-1.asm текст программы из листинга 6.1. 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 sprintLF в регистр еах должен быть записан адрес, необходимо использовать дополнительную переменную. Для этого запишем значение регистра еах в переменную buf1 (mov [buf1], еах), а затем запишем адрес переменной buf1 в регистр еах (mov еах, buf1) и вызовем функцию sprintLF.

```

1 %include 'in_out.asm'
2
3 SECTION .bss
4 buf1: RESB 80
5
6 SECTION .text
7 GLOBAL _start
8 _start:
9
10 mov eax,'6'
11 mov ebx,'4'
12 add eax,ebx
13 mov [buf1],eax
14 mov eax,buf1
15 call sprintf
16
17 call quit

```

Рис. 2.2: Рис 2.1.2: Демонстрация текста программы в файле

Создадим исполняемый файл и проверим его

```

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

```

Рис. 2.3: Рис 2.1.3: Создание исполняемого файла и проверка работы

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),

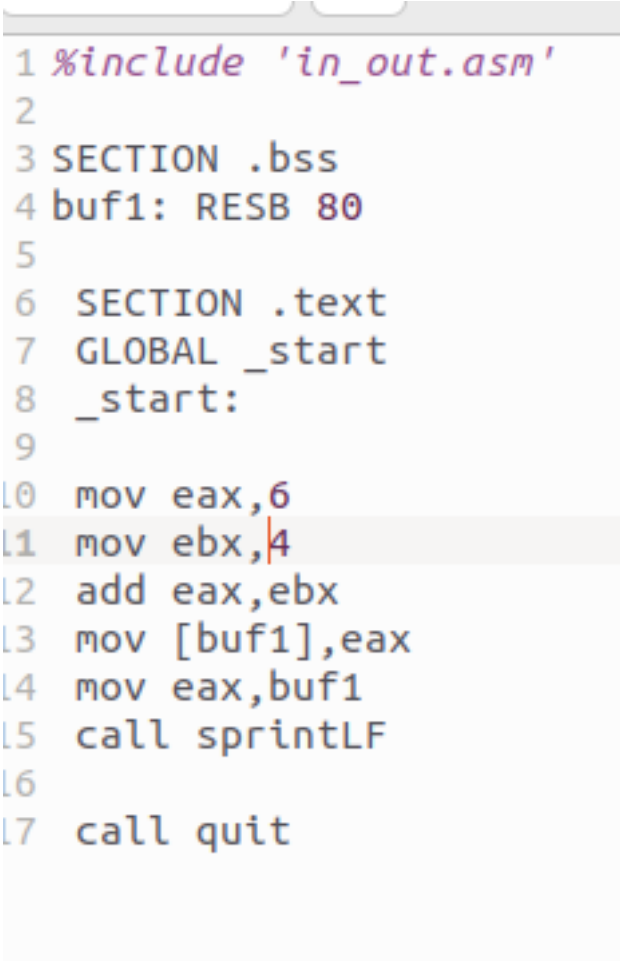
а код символа 4 – 00110100 (52). Команда `add eax,ebx` запишет в регистр `eax` сумму кодов – 01101010 (106), что в свою очередь является кодом символа `j` (см. таблицу ASCII в приложении).

Далее изменим текст программы и вместо символов, запишем в регистры числа. Исправим текст программы следующим образом:

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

на строки

`mov eax,6 mov ebx,4`



```
1 %include 'in_out.asm'
2
3 SECTION .bss
4 buf1: RESB 80
5
6 SECTION .text
7 GLOBAL _start
8 _start:
9
10 mov eax,6
11 mov ebx,4
12 add eax,ebx
13 mov [buf1],eax
14 mov eax,buf1
15 call sprintf
16
17 call quit
```

Рис. 2.4: Рис 2.1.4: Измененная программа

Создадим исполняемый файл и запустим его.


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

perfilov@akperfilov:~/work/arch-pc/lab06$
```

Рис. 2.5: Рис 2.1.5: Создание исполняемого файла и проверка работы

Как и в предыдущем случае при исполнении программы мы не получим число 10. В данном случае выводится символ с кодом 10. Пользуясь таблицей ASCII определим какому символу соответствует код 10, это LF, . Который не отображается на выводе 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 6.1 с использованием этих функций.

Создадим файл lab6-2.asm в каталоге ~/work/arch-pc/lab06 и введем в него текст программы из листинга 6.2.

```
perfilov@akperfilov:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-2.asm
```

Рис. 2.6: Рис 2.1.6: Создание файла

```

1 %include 'in_out.asm'
2
3 SECTION .text
4 GLOBAL _start
5 _start:
6
7 mov eax, '6'
8 mov ebx, '4'
9 add eax, ebx
10 call iprintLF
11
12 call quit

```

Рис. 2.7: Рис 2.1.7: Программа

Создадим исполняемый файл и запустим его.

```

perfilov@akperfilov:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
perfilov@akperfilov:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
perfilov@akperfilov:~/work/arch-pc/lab06$ ./lab6-2
106
perfilov@akperfilov:~/work/arch-pc/lab06$

```

Рис. 2.8: Рис 2.1.8: Создание исполняемого файла и проверка работы

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 отличие от программы из листинга 6.1, функция `iprintLF` позволяет вывести число, а не символ, кодом которого является это число.

Аналогично предыдущему примеру изменим символы на числа. Заменяем стро-

КИ

- `mov eax,'6'`
- `mov ebx,'4'`

на строки

- `mov eax,6`
- `mov ebx,4`

```
1 %include 'in_out.asm'
2
3 SECTION .text
4 GLOBAL _start
5 _start:
6
7 mov eax,6
8 mov ebx,4
9 add eax,ebx
10 call iprintLF
11
12 call quit
```

Рис. 2.9: Рис 2.1.9: Демонстрация измененной программы

Создадим исполняемый файл и запустим его.

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

Рис. 2.10: Рис 2.1.10: Создание исполняемого файла и проверка работы

В результате мы получили число 10

Заменяем функцию `iprintLF` на `iprint`. Создадим исполняемый файл и запустим его.

```
1 %include 'in_out.asm'
2
3 SECTION .text
4 GLOBAL _start
5 _start:
6
7 mov eax,6
8 mov ebx,4
9 add eax,ebx
10 call iprint
11
12 call quit
```

Рис. 2.11: Рис 2.1.11: Именованная программа

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

Рис. 2.12: Рис 2.1.12: Создание исполняемого файла и проверка работы

В результате было получено число 10, но в данном случае `iprint` число выведено без красной строки

Выполнение арифметических операций в NASM


В качестве примера выполнения арифметических операций в NASM приведем программу вычисления арифметического выражения $f(x) = (5 * 2 + 3) / 3$.

Создадим файл `lab6-3.asm` в каталоге `~/work/arch-pc/lab06`:

```
10perfilov@akperfilov:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/lab6-3.asm
perfilov@akperfilov:~/work/arch-pc/lab06$
```

Рис. 2.13: Рис 2.2.1: Создание файла

Внимательно изучим текст программы из листинга 6.3 и введем в lab6-3.asm.



```
1 %include 'in_out.asm' ; подключение внешнего файла
2
3 SECTION .data
4
5 div: DB 'Результат: ',0
6 rem: DB 'Остаток от деления: ',0
7
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 ; ---- Вычисление выражения
13 mov eax,5 ; EAX=5
14 mov ebx,2 ; EBX=2
15 mul ebx ; EAX=EAX*EBX
16 add eax,3 ; EAX=EAX+3
17 xor edx,edx ; обнуляем EDX для корректной работы div
18 mov ebx,3 ; EBX=3
19 div ebx ; EAX=EAX/3, EDX=остаток от деления
20
21 mov edi,eax ; запись результата вычисления в 'edi'
22
23 ; ---- Вывод результата на экран
24
25 mov eax,div ; вызов подпрограммы печати
26 call sprint ; сообщения 'Результат: '
27 mov eax,edi ; вызов подпрограммы печати значения
28 call iprintLF ; из 'edi' в виде символов
29 mov eax,rem ; вызов подпрограммы печати
30 call sprint ; сообщения 'Остаток от деления: '
31 mov eax,edx ; вызов подпрограммы печати значения
32 call iprintLF ; из 'edx' (остаток) в виде символов
33
34 call quit ; вызов подпрограммы завершения
```

Рис. 2.14: Рис 2.2.2: Программа

Создадим исполняемый файл и запустим его.

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

Рис. 2.15: Рис 2.2.3: Создание исполняемого файла и проверка работы

Изменим текст программы для вычисления выражения $f(x) = (4 * 6 + 2)/5$.

```

1 %include 'in_out.asm' ; подключение внешнего файла
2
3 SECTION .data
4
5 div: DB 'Результат: ',0
6 rem: DB 'Остаток от деления: ',0
7
8 SECTION .text
9 GLOBAL _start
10 _start:
11
12 ; ---- Вычисление выражения
13 mov eax,4 ; EAX=4
14 mov ebx,6 ; EBX=6
15 mul ebx ; EAX=EAX*EBX
16 add eax,2 ; EAX=EAX+2
17 xor edx,edx ; обнуляем EDX для корректной работы div
18 mov ebx,5 ; EBX=5
19 div ebx ; EAX=EAX/5, EDX=остаток от деления
20
21 mov edi,eax ; запись результата вычисления в 'edi'
22
23 ; ---- Вывод результата на экран
24
25 mov eax,div ; вызов подпрограммы печати
26 call sprint ; сообщения 'Результат: '
27 mov eax,edi ; вызов подпрограммы печати значения
28 call iprintLF ; из 'edi' в виде символов
29 mov eax,rem ; вызов подпрограммы печати
30 call sprint ; сообщения 'Остаток от деления: '
31 mov eax,edx ; вызов подпрограммы печати значения
32 call iprintLF ; из 'edx' (остаток) в виде символов
33
34 call quit ; вызов подпрограммы завершения

```

Рис. 2.16: Рис 2.2.4: Программа

Создадим исполняемый файл и проверим его работу.

```

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

```

Рис. 2.17: Рис 2.2.5: Создание исполняемого файла и проверка работы

В качестве другого примера рассмотрим программу вычисления варианта задания по номеру студенческого билета, работающую по следующему алгоритму:

вывести запрос на введение № студенческого билета вычислить номер варианта по формуле: $(S_n \bmod 20) + 1$, где S_n – номер студенческого билета (В данном случае $a \bmod b$ - это остаток от деления a на b). вывести на экран номер варианта.

В данном случае число, над которым необходимо проводить арифметические операции, вводится с клавиатуры. Как отмечалось выше ввод с клавиатуры осуществляется в символьном виде и для корректной работы арифметических операций в NASM символы необходимо преобразовать в числа. Для этого может быть использована функция `atoi` из файла `in_out.asm`.

Создадим файл `variant.asm` в каталоге `~/work/arch-pc/lab06`:

```

perfilov@akperfilov:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/variant.asm
perfilov@akperfilov:~/work/arch-pc/lab06$

```

Рис. 2.18: Рис 2.2.6: Создание файла

Внимательно изучим текст программы из листинга 6.4 и введем в файл `variant.asm`.

lab6-1.asm	×	lab6-2.asm	×
1		<code>%include 'in_out.asm'</code>	
2			
3		<code>SECTION .data</code>	
4		<code>msg: DB 'Введите № студенческого билета: ',0</code>	
5		<code>rem: DB 'Ваш вариант: ',0</code>	
6			
7		<code>SECTION .bss</code>	
8		<code>x: RESB 80</code>	
9			
10		<code>SECTION .text</code>	
11		<code>GLOBAL _start</code>	
12		<code>_start:</code>	
13			
14		<code>mov eax, msg</code>	
15		<code>call sprintLF</code>	
16			
17		<code>mov ecx, x</code>	
18		<code>mov edx, 80</code>	
19		<code>call sread</code>	
20			
21		<code>mov eax,x ; вызов подпрограммы преобразования</code>	
22		<code>call atoi ; ASCII кода в число, `eax=x</code>	
23		<code>xor edx,edx</code>	
24		<code>mov ebx,20</code>	
25		<code>div ebx</code>	
26		<code>inc edx</code>	
27			
28		<code>mov eax,rem</code>	
29		<code>call sprint</code>	
30		<code>mov eax,edx</code>	
31		<code>call iprintLF</code>	
32			
33		<code>call quit</code>	

Рис. 2.19: Рис 2.2.7: Программа

Создадим исполняемый файл и запустим его.

```
perfilov@akperfilov:~/work/arch-pc/lab06$ nasm -f elf variant.asm
perfilov@akperfilov:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
perfilov@akperfilov:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132231438
Ваш вариант: 19
```

Рис. 2.20: Рис 2.2.8: Создание исполняемого файла и проверка работы

Вопросы из лаб. работы

Включите в отчет по выполнению лабораторной работы ответы на следующие вопросы:

- 1) Какие строки листинга 6.4 отвечают за вывод на экран сообщения 'Ваш вариант:'?

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

- 2) Для чего используются следующие инструкции?

```
mov esx, x - перемещает адрес вводимой строки в esx
mov edx, 80 - записывает длину строки в регистр edx
call sread - вызывает подпрограммы, которые обеспечивают ввод сообщения с помощью клавиатуры
```

- 3) Для чего используется инструкция "call atoi"?

Она используется для вызова подпрограммы, которая преобразует ASCII код символа в целое число, записывая его в результат регистра eax

- 4) Какие строки листинга 6.4 отвечают за вычисления варианта?

```
xor edx, edx - обнуление ebx для div
mov ebx, 10 - ebx=10
div ebx - eax = eax/10,
edx - остаток от деления
inc edx - edx=edx+1
```

- 5) В какой регистр записывается остаток от деления при выполнении инструкции "div ebx"?

При `div ebx` остаток от деления записывается в `edx`

6) Для чего используется инструкция `inc edx`?

`inc edx` увеличивает значение регистра `edx` на +1

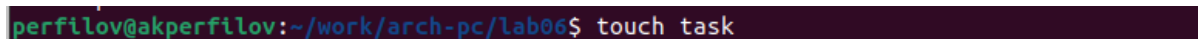
7) Какие строки листинга 6.4 отвечают за вывод на экран результата вычисления?

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

3 Самостоятельная работа

Задание№1 Написать программу вычисления выражения $y=f(x)$. Программа должна выводить выражение для вычисления, выводить запрос на ввод значения x , вычислять заданное выражение в зависимости от введенного x , выводить результат вычислений. Вид функции $f(x)$ выбрать из таблицы 6.3 вариантов заданий в соответствии с номером полученным при выполнении лабораторной работы. Создайте исполняемый файл и проверьте его работу для значений x_1 и x_2 из 6.3. При выполнении задания преобразовывать (упрощать) выражения для $f(x)$ нельзя. При выполнении деления в качестве результата можно использовать только целую часть от деления и не учитывать остаток (т.е. $5 : 2 = 2$)

Создадим новый файл для задания и напишем программу для $f(x)=((1/3)x+5)*7$



```
perfilov@akperfilov:~/work/arch-pc/lab06$ touch task
```

Рис. 3.1: Рис 3.1.1: Создание файла

```

1 %include 'in_out.asm'
2
3 SECTION .data
4     msg: db 'Введите x: ', 0
5     rem: db 'Результат: ', 0
6 SECTION .bss
7 x: RESB 80
8 SECTION .text
9 GLOBAL _start
10
11 _start:
12
13     mov eax, msg
14     call sprintLF
15
16     mov ecx, x
17     mov edx, 80
18     call sread
19     mov eax, x
20     call atoi
21
22     mov ebx, 1
23     mul ebx
24     xor edx,edx
25     mov ebx, 3
26     div ebx
27
28     add eax, 5
29     mov ecx, 7
30     mul ecx
31     mov edi, eax
32
33     mov eax, rem
34     call sprint
35     mov eax, edi
36     call iprintLF
37
38     call quit
39

```

Рис. 3.2: Рис 3.1.2: Программа

Проверим программу

```
perfilov@akperfilov:~/work/arch-pc/lab06$ nasm -f elf task.asm
perfilov@akperfilov:~/work/arch-pc/lab06$ ld -m elf_i386 -o task task.o
perfilov@akperfilov:~/work/arch-pc/lab06$ ./task
Введите x:
3
Результат: 42
perfilov@akperfilov:~/work/arch-pc/lab06$ ./task
Введите x:
9
Результат: 56
perfilov@akperfilov:~/work/arch-pc/lab06$
```

Рис. 3.3: Рис 3.1.3: Проверка программы

Загрузим все файлы на github

4 Выводы

Я освоил арифметические инструкции языка ассемблера NASM вместе с практикой