

Отчёт по лабораторной работе №6

Арифметические операции в NASM

Лань Цяньин

1. Цель работы

Освоение арифметических инструкций языка ассемблера NASM

2. Порядок выполнения лабораторной работы

2.1. Символьные и численные данные в NASM

1. Создан каталог для программ лабораторной работы № 6, выполнен переход в него и создан файл lab6-1.asm с помощью команд mkdir, cd и touch (рис. Рисунок 1).

```
clanj1@clanj1:~/Desktop$ mkdir ~ /work/arch-pc/lab06
clanj1@clanj1:~/Desktop$ cd ~ /work/arch-pc/lab06
clanj1@clanj1:~/work/arch-pc/lab06$ touch lab6-1.asm
clanj1@clanj1:~/work/arch-pc/lab06$ gedit lab6-1.asm
```

Рисунок 1: Создание каталога lab06 и файла lab6-1.asm

В результате был создан каталог lab06, в котором подготовлен файл lab6-1.asm для дальнейшей работы над программой.

2. В файл lab6-1.asm был введён текст программы из листинга 6.1 (рис. Рисунок 2).

The screenshot shows a text editor window titled "lab6-1.asm" with the file path "~/work/arch-pc/lab06". The code is as follows:

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

Рисунок 2: Ввод текста программы lab6-1.asm

Дан пример работы с регистрами и подпрограммой `sprintLF`: после вычислений значение сохраняется в `buf1`, затем в `eax` загружается адрес `buf1` и выполняется вывод.

Файл `in_out.asm` был скопирован в каталог `lab06`, после чего выполнены сборка и запуск программы. (рис. Рисунок 3)

```
clanj1@clanj1:~/work/arch-pc/lab06$ cp /media/sf_VB_Share/in_out.asm ~/work/arch-pc/Lab06
clanj1@clanj1:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
clanj1@clanj1:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
clanj1@clanj1:~/work/arch-pc/lab06$ ./lab-1
bash: ./lab-1: No such file or directory
clanj1@clanj1:~/work/arch-pc/lab06$ ./lab6-1
j
```

Рисунок 3: Сборка и запуск программы lab6-1

В результате на экране был выведен символ `j`, что связано с тем, что в программе выполняется сложение кодов символов '`6`' и '`4`', а полученное значение соответствует символу `j` в таблице ASCII.

3. Текст программы `lab6-1.asm` был изменён: вместо символьных констант в регистры записаны числовые значения.

Строки

`mov eax, '6'` и `mov ebx, '4'`

заменены на

`mov eax, 6` и `mov ebx, 4` в соответствии с заданием.(рис. Рисунок 4)

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

Рисунок 4: Изменение текста программы lab6-1.asm

Программа была скомпилирована, слинкована и запущена командами nasm, ld и ./lab6-1 после замены символьных значений на числовые в файле lab6-1.asm. (рис. Рисунок 5)

```
clanj1@clanj1:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
clanj1@clanj1:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
clanj1@clanj1:~/work/arch-pc/lab06$ ./lab6-1
```

Рисунок 5: Сборка и запуск программы lab6-1

После запуска программы на экране не появляется видимого вывода, так как функция sprintLF интерпретирует значение в регистре eax как ASCII-код символа.

При значении eax = 10 выводится управляемый символ с кодом 10 (перевод строки), поэтому отображаемый результат визуально отсутствует.

4. По заданию создан файл lab6-2.asm (листинг 6.2): в регистры eax и ebx записываются символы '6' и '4', выполняется сложение add eax, ebx, после чего результат выводится подпрограммой iprintLF. (рис. Рисунок 6)

The screenshot shows a text editor window with the title bar "lab6-2.asm" and the path "/work/arch-pc/lab06". The code in the editor is:

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

Рисунок 6: Редактирование файла lab6-2.asm (листинг 6.2)

```
clanj1@clanj1:~/work/arch-pc/lab06$ touch lab6-2.asm
clanj1@clanj1:~/work/arch-pc/lab06$ gedit lab6-2.asm
clanj1@clanj1:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
clanj1@clanj1:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
clanj1@clanj1:~/work/arch-pc/lab06$ ./lab6-2
106
```

Рисунок 7: Сборка и запуск программы lab6-2, вывод результата

После запуска программа выводит 106, потому что команда add складывает ASCII-коды символов '6' и '4' ($54 + 52 = 106$). Подпрограмма iprintLF выводит содержимое eax как десятичное число, поэтому на экране отображается именно 106. (рис. Рисунок 7)

5. В файле lab6-2.asm символы заменены на числа, а iprintLF – на iprint:
mov eax, '6' → mov eax, 6, mov ebx, '4' → mov ebx, 4; вызов call iprint.
(рис. Рисунок 8)

The screenshot shows a text editor window with the title bar "lab6-2.asm" and the path "/work/arch-pc/lab06". The code in the editor is:

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

Рисунок 8: Редактирование lab6-2.asm: числа и iprint

```

clanj1@clanj1:~/work/arch-pc/lab06$ gedit lab6-2.asm
clanj1@clanj1:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
clanj1@clanj1:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
clanj1@clanj1:~/work/arch-pc/lab06$ ./lab6-2
10clanj1@clanj1:~/work/arch-pc/lab06$
```

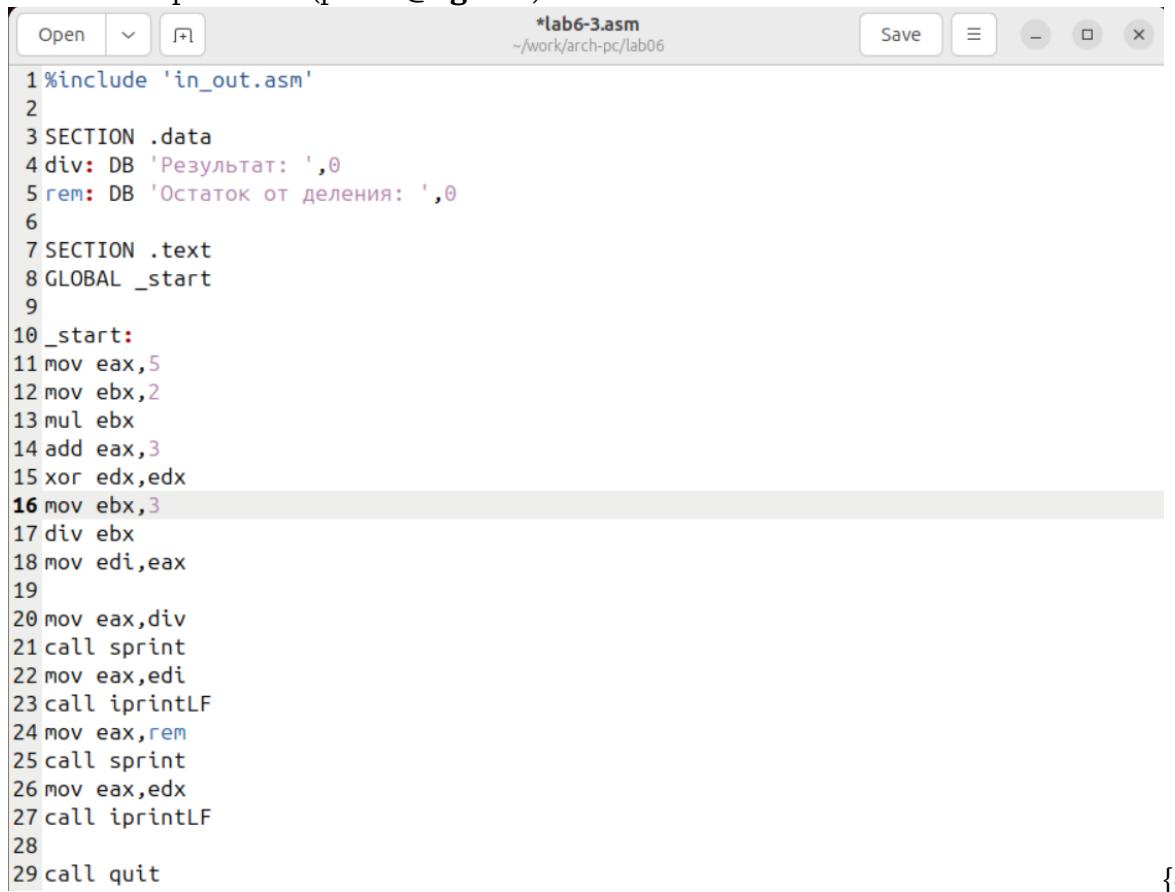
Рисунок 9: Сборка, линковка и запуск lab6-2: вывод 10 без переноса строки

При выполнении программы выводится число 10, так как складываются значения 6 и 4.

Функция `iprint` печатает число без перевода строки, тогда как `iprintLF` добавляет перенос строки. (рис. Рисунок 9)

2.2. Выполнение арифметических операций в NASM

6. Создан файл `lab6-3.asm` и введён текст программы из листинга 6.3 для вычисления выражения (рис. ?@fig-010)



```

1 %include 'in_out.asm'
2
3 SECTION .data
4 div: DB 'Результат: ',0
5 rem: DB 'Остаток от деления: ',0
6
7 SECTION .text
8 GLOBAL _start
9
10 _start:
11 mov eax,5
12 mov ebx,2
13 mul ebx
14 add eax,3
15 xor edx,edx
16 mov ebx,3
17 div ebx
18 mov edi, eax
19
20 mov eax,div
21 call sprint
22 mov eax,edi
23 call iprintLF
24 mov eax,rem
25 call sprint
26 mov eax,edx
27 call iprintLF
28
29 call quit
```

{#fig-

010-width=80%}

Программа была скомпилирована, слинкована и запущена командами nasm, ld и ./lab6-3. (рис. Рисунок 10)

```
clanj1@clanj1:~/work/arch-pc/lab06$ touch lab6-3.asm
clanj1@clanj1:~/work/arch-pc/lab06$ gedit lab6-3.asm
clanj1@clanj1:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
clanj1@clanj1:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
clanj1@clanj1:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
```

Рисунок 10: Сборка и запуск программы lab6-3: результат и остаток

В результате работы программы получено: Результат: 4, Остаток от деления: 1,

что соответствует вычислению $(13 / 3 = 4)$ и остатку (1).

Для задания 6.3.2 был изменён текст программы для вычисления выражения

$(f(x) = (4 \cdot 6 + 2)/5)$. В программе изменены константы:

mov eax, 4, mov ebx, 6, затем mul ebx и add eax, 2.

Перед делением выполняется xor edx, edx, после чего деление выполняется на 5: mov ebx, 5 и div ebx.

После div частное находится в eax, остаток — в edx. (рис. Рисунок 11)

```

1 %include 'in_out.asm'
2
3 SECTION .data
4 div: DB 'Результат: ',0
5 rem: DB 'Остаток от деления: ',0
6
7 SECTION .text
8 GLOBAL _start
9
10 _start:
11 mov eax,4
12 mov ebx,6
13 mul ebx
14 add eax,2
15 xor edx,edx
16 mov ebx,5
17 div ebx
18 mov edi, eax
19
20 mov eax,div
21 call sprint
22 mov eax,edi
23 call iprintLF
24 mov eax,rem
25 call sprint
26 mov eax,edx
27 call iprintLF
28
29 call quit

```

Рисунок 11: Изменённый текст программы lab6-3.asm для вычисления $(4 \cdot 6 + 2) / 5$

Программа была скомпилирована (nasm), слинкована (ld) и запущена.

В результате получено: Результат: 5, Остаток от деления: 1.

Это соответствует вычислению $((4 \cdot 6 + 2) = 26)$, $(26 / 5 = 5)$ и остаток (1). (рис. Рисунок 12)

```

clanj1@clanj1:~/work/arch-pc/lab06$ gedit lab6-3.asm
clanj1@clanj1:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
clanj1@clanj1:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
clanj1@clanj1:~/work/arch-pc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1

```

Рисунок 12: Сборка и запуск программы: вывод результата и остатка

7. По Листингу 6.4 реализована программа variant.asm для вычисления номера варианта по номеру студенческого билета по формуле $(S_n \bmod 20) + 1$ (рис. Рисунок 13).

The screenshot shows a text editor window with the file name "variant.asm" at the top. The code is as follows:

```
1 %include 'in_out.asm'
2
3 SECTION .data
4 msg: DB "Введите № студенческого билета: ',0
5 rem: DB "Ваш вариант: ',0
6
7 SECTION .bss
8 x: RESB 80
9
10 SECTION .text
11 GLOBAL _start
12
13 _start:
14 mov eax, msg
15 call sprintLF
16 mov ecx, x
17 mov edx, 80
18 call sread
19 mov eax,x
20 call atoi
21 xor edx,edx
22 mov ebx,20
23 div ebx
24 inc edx
25 mov eax,rem
26 call sprint
27 mov eax,edx
28 call iprintLF
29 call quits|
```

Рисунок 13: Исходный код программы variant.asm по Листингу 6.4

Программа была скомпилирована с помощью nasm, слинкована с помощью ld и запущена. В качестве входных данных был введён номер студенческого билета 1132254528. В результате работы программы на экран был выведен номер варианта: 9. (рис. Рисунок 14)

```
clanj1@clanj1:~/work/arch-pc/lab06$ touch variant.asm
clanj1@clanj1:~/work/arch-pc/lab06$ gedit variant.asm
clanj1@clanj1:~/work/arch-pc/lab06$ nasm -f elf variant.asm
clanj1@clanj1:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
clanj1@clanj1:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132254528
Ваш вариант: 9
```

Рисунок 14: Сборка и запуск программы variant

Ответы на вопросы к листингу 6.4

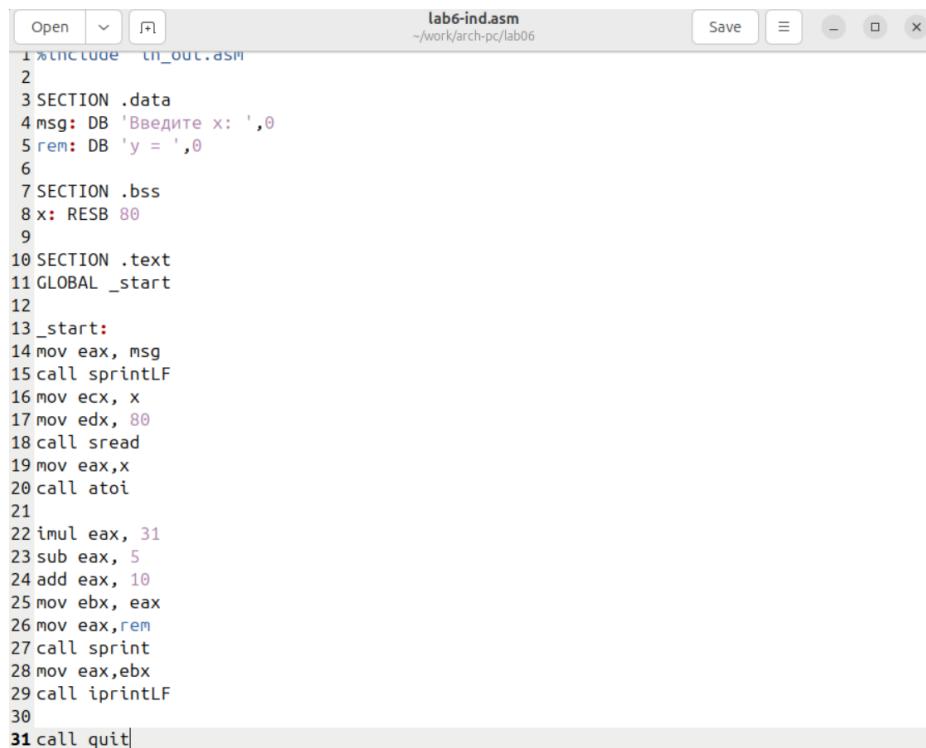
1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения «Ваш вариант:»? Ответ: За вывод сообщения «Ваш вариант:» отвечают строки,

в которых в регистр eax загружается адрес строки с текстом сообщения, после чего вызывается подпрограмма вывода sprint (или sprintLF)

2. Для чего используются следующие инструкции? `mov ecx, x` `mov edx, 80` `call sread` Ответ □ Данные инструкции используются для чтения строки с клавиатуры: ecx содержит адрес буфера x, edx задаёт максимальный размер вводимой строки, подпрограмма sread выполняет ввод данных.
3. Для чего используется инструкция `call atoi`? Ответ □ Инструкция call atoi используется для преобразования введённой строки, содержащей число в текстовом виде, в целочисленное значение.
4. Какие строки листинга 6.4 отвечают за вычисление варианта? Ответ □ За вычисление номера варианта отвечают строки, содержащие арифметические операции, включая инструкцию деления `div`, а также подготовку регистров перед выполнением деления
5. В какой регистр записывается остаток от деления при выполнении инструкции `div ebx`? Ответ □ Остаток от деления при выполнении инструкции `div ebx` записывается в регистр edx.
6. Для чего используется инструкция `inc edx`? Ответ □ Инструкция inc edx используется для увеличения значения регистра edx на единицу, что позволяет скорректировать номер варианта в соответствии с заданным алгоритмом.
7. Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений? Ответ □ За вывод результата вычислений отвечают строки, в которых значение варианта преобразуется в строку и выводится на экран с помощью подпрограммы sprint (или sprintLF).

3. Задание для самостоятельной работы

По образцу из Листингов 6.3–6.4 (ввод x через `sread + atoi`) написана программа `lab6-ind.asm`, которая вычисляет вариант 9 из табл. 6.3: $y = 10 + (31x - 5)$, и выводит результат.



The screenshot shows a text editor window with the file `lab6-ind.asm` open. The code is written in assembly language and performs the following steps:

- Defines sections: .data, .bss, .text.
- Creates global variables: `_start`, `msg` (containing "Введите x: "), and `rem` (containing "y = ").
- Allocates memory for `x` in the .bss section.
- Enters the .text section and defines the entry point `_start`.
- Prints the message "Введите x: " using `sprintLF`.
- Reads input from `sread` into `eax`.
- Converts the input string to a number using `atoi`.
- Performs the calculation: $y = 10 + (31 \times eax - 5)$.
- Prints the result using `sprint`.
- Prints a new line using `iprintLF`.
- Exits the program using `quit`.

```
Open ▾ ln_out.asm lab6-ind.asm ~/work/arch-pc/lab06 Save ⌂ ×
1 %INCLUDE ln_out.asm
2
3 SECTION .data
4 msg: DB 'Введите x: ',0
5 rem: DB 'y = ',0
6
7 SECTION .bss
8 x: RESB 80
9
10 SECTION .text
11 GLOBAL _start
12
13 _start:
14 mov eax, msg
15 call sprintLF
16 mov ecx, x
17 mov edx, 80
18 call sread
19 mov eax,x
20 call atoi
21
22 imul eax, 31
23 sub eax, 5
24 add eax, 10
25 mov ebx, eax
26 mov eax,rem
27 call sprint
28 mov eax,ebx
29 call iprintLF
30
31 call quit|
```

Рисунок 1: Код программы `lab6-ind.asm` (ввод x, вычисление y)

```
clanj1@clanj1:~/work/arch-pc/lab06$ gedit lab6-ind.asm
clanj1@clanj1:~/work/arch-pc/lab06$ nasm -f elf lab6-ind.asm
clanj1@clanj1:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-ind lab6-ind.o
clanj1@clanj1:~/work/arch-pc/lab06$ ./lab6-ind
Ведите x:
3
y = 98
clanj1@clanj1:~/work/arch-pc/lab06$ ./lab6-ind
Ведите x:
1
y = 36
```

Рисунок 2: Запуск программы и проверка для $x=3$ и $x=1$

Проверка: при $x=3$ получено $y=98$, при $x=1$ получено $y=36$ — соответствует формуле варианта 9.

Выводы

В ходе выполнения лабораторной работы №6 были изучены арифметические инструкции языка ассемблера NASM и принципы работы с числовыми данными. Была реализована программа вычисления номера варианта и программа вычисления заданного арифметического выражения. Цель лабораторной работы достигнута.