

Отчёт по лабораторной работе №6

Арифметические операции в NASM.

Грачева Мария Валерьевна

Содержание

1	Цель работы	5
2	Теоретическое введение	6
3	Выполнение лабораторной работы	7
4	Ответы на вопросы	14
5	Самостоятельная работа	16
6	Выводы	19
	Список литературы	20

Список иллюстраций

3.1	Создание каталога	7
3.2	Введение текста	7
3.3	Создание исполняемого файла	7
3.4	Запуск файла	8
3.5	Изменение текста	8
3.6	Запуск файла 2	8
3.7	Создание файла lab06-2	9
3.8	Введение текста 2	9
3.9	Запуск файла 3	9
3.10	Замена символов	10
3.11	Запуск файла 4	10
3.12	Замена iprintLF на iprint	11
3.13	Запуск файла 5	11
3.14	Создание файла lab06-3	12
3.15	Запуск файла 6	12
3.16	Изменение текста 2	12
3.17	Запуск файла 7	12
3.18	Создание файла variant.asm	12
3.19	Программа для вычисление варианта	13
3.20	Получение варианта	13
5.1	Создание файла task.asm	16
5.2	Написание программы	17
5.3	Проверка для первого значения - 3	18
5.4	Проверка для второго значения - 1	18

Список таблиц

1 Цель работы

Освоение арифметических инструкций языка ассемблера NASM

2 Теоретическое введение

Большинство инструкций на языке ассемблера требуют обработки операндов. Адрес операнда 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 команде, Например: `mov ax,2`.
- *Адресация памяти* – операнд задает адрес в памяти. В команде указывается символическое обозначение ячейки памяти, над содержимым которой требуется выполнить операцию.

Арифметические операции в NASM

Сложение - `add` Вычитание - `sub` Умножение - `mul` Деление - `div`

Для выполнения лабораторных работ в файле `in_out.asm` реализованы подпрограммы для преобразования ASCII символов в числа и обратно. Это:

- *iprint* – вывод на экран чисел в формате ASCII, перед вызовом `iprint` в регистр `eax` необходимо записать выводимое число (`mov eax,`).
- *iprintLF* – работает аналогично `iprint`, но при выводе на экран после числа добавляет к символ перевода строки.
- *atoi* – функция преобразует `ascii`-код символа в целое число и записывает результат в регистр `eax`, перед вызовом `atoi` в регистр `eax` необходимо записать число (`mov eax,`).

3 Выполнение лабораторной работы

Создаю каталог для программ лабораторной работы № 6, перехожу в него и создаю файл lab6-1.asm(рис. 3.1).

```
mvgracheva@dk5n56 ~ $ mkdir ~/work/arch-pc/lab06
mvgracheva@dk5n56 ~ $ cd ~/work/arch-pc/lab06
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ touch lab6-1.asm
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ mc
```

Рис. 3.1: Создание каталога

Ввожу текст для программы вывода значения регистра eax(рис. 3.2).

```
lab6-1.asm [----] 9 L:[ 1+12 13/ 13] *(172 / 172b) <EOF>
#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. 3.2: Введение текста

Создаю исполняемый файл и запускаю его (рис. 3.3), (рис. 3.4).

```
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
```

Рис. 3.3: Создание исполняемого файла

```

mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ ./lab6-1
j
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ █

```

Рис. 3.4: Запуск файла

Заменяю текст программы (рис. 3.5), должны получить символ 49-1 48-0, то есть 4948, но в итоге ничего не вывелось (рис. 3.6).

```

_start:
mov  eax,6
mov  ebx,4
add  eax,ebx
mov  [buf1],eax
mov  eax,buf1

```

Рис. 3.5: Изменение текста

```

mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ ./lab6-1

mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ █

```

Рис. 3.6: Запуск файла 2

Создаю файл lab06-2 (рис. 3.7), ввожу в неё программу (рис. 3.8), создаю исполняемый файл и проверяю работу (рис. 3.9).


```
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ touch lab6-2.asm
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $
```

Рис. 3.7: Создание файла lab06-2

```
lab6-2.asm [-M--] 9 L:[ 1+
#include 'in_out.asm'
SECTION .text
GLOBAL _start
_start:
mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF
call quit
```

Рис. 3.8: Введение текста 2

```
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ ./lab6-2
106
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $
```

Рис. 3.9: Запуск файла 3

Меняю символы на числа (рис. 3.10).

```

GLOBAL _start
_start:
mov  eax,6
mov  ebx,4
add  eax,ebx
call iprintLF
call quit

```

Рис. 3.10: Замена символов

Создаю исполняемый файл. Получаем 10, программа работает верно! (рис. 3.11).

```

mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ ./lab6-2
10
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ █

```

Рис. 3.11: Запуск файла 4

Заменяю `iprintLF` на `iprint` (рис. 3.12), создаю исполняемый файл и проверяю работу. Отличие заключается в том, что нет перехода на новую строку после завершения программы (рис. 3.13).

```
GLOBAL _start
_start:
mov  eax,6
mov  ebx,4
add  eax,ebx
call iprint
call quit
```

Рис. 3.12: Замена iprintLF на iprint

```
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ ./lab6-2
10mvgracheva@dk5n56 ~/work/arch-pc/lab06 $
```

Рис. 3.13: Запуск файла 5

Создаю файл lab6-3.asm (рис. 3.14), ввожу текст и проверяю работу (рис. 3.15).

```
10mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ touch lab6-3.asm
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $
```

Рис. 3.14: Создание файла lab06-3

```
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $
```

Рис. 3.15: Запуск файла 6

Изменяю текст программы для вычисления выражения $\boxtimes(\boxtimes) = (4 \boxtimes 6 + 2)/5$ (рис. 3.16). Создаю исполняемый файл и проверяю его работу (рис. 3.17).

```
; --- 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/3, EDX=остаток от деления
```

Рис. 3.16: Изменение текста 2

```
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $
```

Рис. 3.17: Запуск файла 7

Создаю файл (рис. 3.18), ввожу текст программы (рис. 3.19), проверяю работу. Получился 16 вариант (рис. 3.20).

```
Остаток от деления: 1
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ touch variant.asm
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $
```

Рис. 3.18: Создание файла variant.asm

```

task.asm      [----] 10 L: [ 1+27 28/ 28] *(471 / 471b) <EOF>
;-----
; Программа вычисления варианта
;-----
#include "in_out.asm"
SECTION .data
msg: DB "Введите число: ",0
rem: DB "Ответ: ",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
call atoi
mov ebx, 10
mul ebx
sub eax, 5
mov ebx, [eax]
mul ebx
mov eax, rem
call sprint
call iprintLF
call quit
1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перезагрузить 7Поиск 8Удалить

```

Рис. 3.19: Программа для вычисление варианта

```

mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ ./variant
Введите No студенческого билета:
1132226475
Ваш вариант: 16
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $

```

Рис. 3.20: Получение варианта

4 Ответы на вопросы

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

```
msg: DB 'Введите No студенческого билета:',0
```

```
и
```

```
mov eax, msg call sprintLF
```

2. Для чего используются следующие инструкции? `mov ecx, x` `mov edx, 80` `call sread`

Для того, чтобы прочитать то значение `x`, которое ввёл пользователь

3. Для чего используется инструкция "call atoi"?

Для преобразования ASCII кода в число

4. Какие строки листинга 6.4 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"?

Для того, чтобы занести в регистр `edx` остаток от деления

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

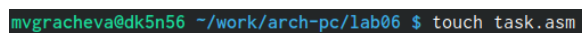
```
rem: DB 'Ваш вариант:',0
```

и

```
mov eax,rem call sprint mov eax,edx call iprintLF
```

5 Самостоятельная работа

Создаю файл task.asm (рис. 5.1), ввожу программу (рис. 5.2), ввожу первое значение - 3 (рис. 5.3), ввожу второе значение - 1 (рис. 5.4).

A terminal window with a dark background. The prompt is 'mvgracheva@dk5n56 ~/work/arch-pc/lab06 \$'. The command 'touch task.asm' is entered and executed.

```
mvgracheva@dk5n56 ~/work/arch-pc/lab06 $ touch task.asm
```

Рис. 5.1: Создание файла task.asm


```

task.asm      [----] 11 L:[ 1+15 1
;-----
; Программа вычисления варианта
;-----
#include 'in_out.asm'
SECTION .data
msg: DB 'Введите число: ',0
rem: DB 'Ответ: ',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
call atoi
mov ebx, 10
mul ebx
sub eax, 5
mov ebx, eax
mul ebx
mov edi, eax
mov eax, rem
call sprintf
mov eax, edi
call iprintLF
call quit

```

Рис. 5.2: Написание программы

```
mvgracheva@dk1n22 ~/work/arch-pc/lab06 $ nasm -f elf task.asm
mvgracheva@dk1n22 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o task task.o
mvgracheva@dk1n22 ~/work/arch-pc/lab06 $ ./task
Введите число:
3
Ответ:
625
mvgracheva@dk1n22 ~/work/arch-pc/lab06 $ mc
```

Рис. 5.3: Проверка для первого значения - 3

```
mvgracheva@dk1n22 ~/work/arch-pc/lab06 $ ./task
Введите число:
1
Ответ:
25
mvgracheva@dk1n22 ~/work/arch-pc/lab06 $
```

Рис. 5.4: Проверка для второго значения - 1

6 Выводы

Освоила арифметические инструкции языка ассемблера NASM

Список литературы