

Отчет по лабораторной работе №6

Дисциплина: архитектура компьютера

Петрова Алевтина Александровна

Содержание

1 Цель работы

Цель данной лабораторной работы - освоение арифметических инструкций языка ассемблера NASM.

2 Задание

1. Символьные и численные данные в NASM
2. Выполнение арифметических операций в NASM
3. Выполнение заданий для самостоятельной работы

3 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. - Регистровая адресация – операнды хранятся в регистрах и в команде используются имена этих реги-

стров, 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 содержанием которой требуется выполнить операцию.

Ввод информации с клавиатуры и вывод её на экран осуществляется в символьном виде. Кодирование этой информации производится согласно кодовой таблице символов ASCII. ASCII – сокращение от American Standard Code for Information Interchange (Американский стандартный код для обмена информацией). Согласно стандарту ASCII каждый символ кодируется одним байтом. Среди инструкций NASM нет такой, которая выводит числа (не в символьном виде). Поэтому, например, чтобы вывести число, надо предварительно преобразовать его цифры в ASCII-коды этих цифр и выводить на экран эти коды, а не само число. Если же выводить число на экран непосредственно, то экран воспримет его не как число, а как последовательность ASCII-символов – каждый байт числа будет воспринят как один ASCII-символ – и выведет на экран эти символы. Аналогичная ситуация происходит и при вводе данных с клавиатуры. Введенные данные будут представлять собой символы, что сделает невозможным получение корректного результата при выполнении над ними арифметических операций. Для решения этой проблемы необходимо проводить преобразование ASCII символов в числа и обратно

4 Выполнение лабораторной работы

4.1 Символьные и численные данные в NASM

С помощью утилиты `mkdir` создаю директорию, в которой буду создавать файлы с программами для лабораторной работы №6 (рис. [1]). Перехожу в созданный каталог с помощью утилиты `cd`.

```
petrovkina1002@fedora ~]$ mkdir ~/work/study/2023-2024/Архитектура\ компьютера/arch-pc/lab06
petrovkina1002@fedora ~]$ cd work/study/2023-2024/Архитектура\ компьютера/arch-pc/lab06
petrovkina1002@fedora lab06$
```

Figure 1: Создание директории

С помощью утилиты touch создаю файл lab6-1.asm (рис. [2]).

```
petrovkina1002@fedora lab06$ touch lab6-1.asm
petrovkina1002@fedora lab06$ ls
lab6-1.asm
petrovkina1002@fedora lab06$
```

Figure 2: Создание файла

Копирую в текущий каталог файл in_out.asm с помощью утилиты cp, т.к. он будет использоваться в других программах (рис. [3]).

```
petrovkina1002@fedora lab06$ cp ~/3арязык/in_out.asm in_out.asm
petrovkina1002@fedora lab06$ ls
in_out.asm lab6-1.asm
petrovkina1002@fedora lab06$
```

Figure 3: Создание копии файла

Открываю созданный файл lab6-1.asm, вставляю в него программу вывода значения регистра eax (рис. [4]).

```
*~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06/lab6-1.asm - Mousepad
Файл Правка Поиск Просмотр Документ Помощь
#include "in_out.asm"
SECTION .bss
buf: resb 80
SECTION .text
GLOBAL _start
_start:
    mov eax, 0
    mov ebx, 4
    add eax, ebx
    mov [buf], eax
    mov esi, buf
    call sprintf
    call exit
```

Figure 4: Редактирование файла

Создаю исполняемый файл программы и запускаю его (рис. [5]). Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.

```
petrovkina1002@fedora lab06$ nasm -f elf lab6-1.asm
petrovkina1002@fedora lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
petrovkina1002@fedora lab06$ ./lab6-1
j
petrovkina1002@fedora lab06$
```

Figure 5: Запуск исполняемого файла

Изменяю в тексте программы символы "6" и "4" на цифры 6 и 4 (рис. [6]).



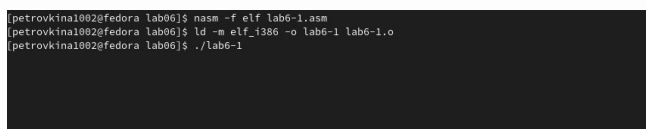
```

#include <io.h>
SECTION .bss
buf: resb 80
SECTION .text
GLOBAL _start
_start:
mov eax, 6
mov ebx, 4
add eax, ebx
mov [buf], eax
mov ecx, buf
call sprintf
call quit

```

Figure 6: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его (рис. [7]). Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.



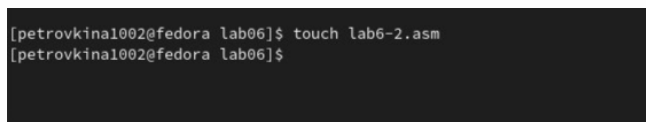
```

[petrovkina1002@fedora lab06]$ nasm -f elf lab6-1.asm
[petrovkina1002@fedora lab06]$ ld -m elf_i386 -o lab6-1 lab6-1.o
[petrovkina1002@fedora lab06]$ ./lab6-1

```

Figure 7: Запуск исполняемого файла

Создаю новый файл lab6-2.asm с помощью утилиты touch (рис. [8]).



```

[petrovkina1002@fedora lab06]$ touch lab6-2.asm
[petrovkina1002@fedora lab06]$

```

Figure 8: Создание файла

Ввожу в файл текст другой программы для вывода значения регистра eax (рис. [9]).



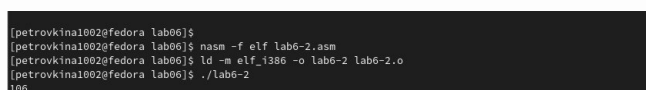
```

#include <io.h>
SECTION .bss
buf: resb 80
SECTION .text
GLOBAL _start
_start:
mov eax, 6
mov ebx, 4
add eax, ebx
mov [buf], eax
mov ecx, buf
call sprintf
call quit

```

Figure 9: Редактирование файла

Создаю и запускаю исполняемый файл lab6-2 (рис. [10]). Теперь вывод число 106, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов "6" и "4".



```

[petrovkina1002@fedora lab06]$
[petrovkina1002@fedora lab06]$ nasm -f elf lab6-2.asm
[petrovkina1002@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[petrovkina1002@fedora lab06]$ ./lab6-2
106

```

Figure 10: Запуск исполняемого файла

Заменяю в тексте программы в файле lab6-2.asm символы "6" и "4" на числа 6 и 4 (рис. [11]).



Figure 11: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. [12]). Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

```
[petrovkina1002@fedora lab06]$ nasm -f elf lab6-2.asm
[petrovkina1002@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[petrovkina1002@fedora lab06]$ ./lab6-2
10
[petrovkina1002@fedora lab06]$
```

Figure 12: Запуск исполняемого файла

Заменяю в тексте программы функцию `iprintLF` на `iprint` (рис. [13]).



Figure 13: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. [14]). Вывод не изменился, потому что символ переноса строки не отображался, когда программа исполнялась с функцией `iprintLF`, а `iprint` не добавляет к выводу символ переноса строки, в отличие от `iprintLF`.

```
[petrovkina1002@fedora lab06]$ 
[petrovkina1002@fedora lab06]$ nasm -f elf lab6-2.asm
[petrovkina1002@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[petrovkina1002@fedora lab06]$ ./lab6-2
10[petrovkina1002@fedora lab06]$
```

Figure 14: Запуск исполняемого файла

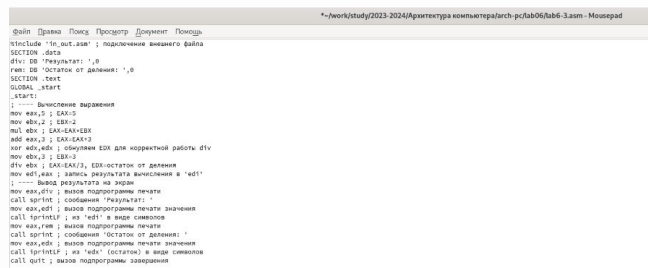
4.2 Выполнение арифметических операций в NASM

Создаю файл `lab6-3.asm` с помощью утилиты `touch` (рис. [15]).

```
[petrovkina1002@fedora lab06]$ nasm -f elf lab6-2.asm
[petrovkina1002@fedora lab06]$ ld -m elf_i386 -o lab6-2 lab6-2.o
[petrovkina1002@fedora lab06]$ ./lab6-2
10[petrovkina1002@fedora lab06]$ touch lab6-3.asm
```

Figure 15: Создание файла

Ввожу в созданный файл текст программы для вычисления значения выражения $f(x) = (5 * 2 + 3)/3$ (рис. [16]).

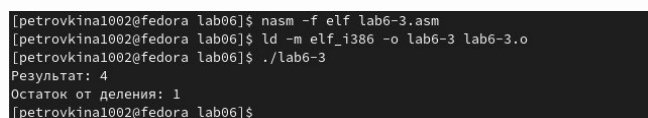


```
*~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06/lab6-3.asm - Mousepad

#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,2 ; EBX=2
mov ebx,2 ; EBX=2
mul ebx ; EAX=EAX*EBX
add eax,3 ; EAX=EAX+3
xor ebx,ebx ; обнуляем EBX для корректной работы div
mov ebx,3 ; EBX=3
div ebx ; EAX=EAX/3, EDI=остаток от деления
mov edi,eax ; запись результата вычисления в "edi"
; ---- Вывод результата на экран
mov eax,div ; вывод подпрограммы печати
call printf ; сообщение "Результат: "
mov eax,rem ; вывод подпрограммы печати
call printf ; из "edi" в виде символов
mov eax,edi ; вывод подпрограммы печати значения
call printf ; сообщение "Остаток от деления: "
mov eax,edi ; вывод подпрограммы печати значения
call printf ; из "edi" (остаток) в виде символов
call quit ; вывод подпрограммы завершения
```

Figure 16: Редактирование файла

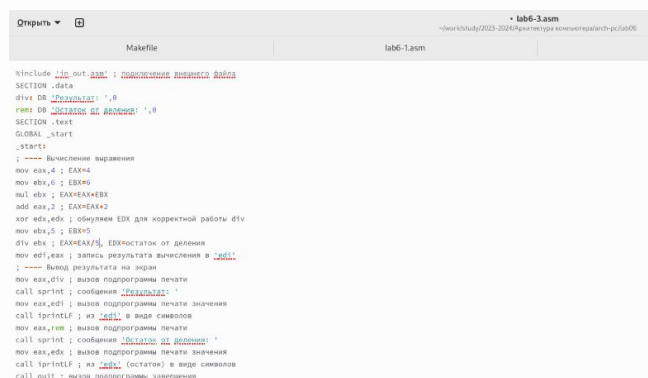
Создаю исполняемый файл и запускаю его (рис. [17]).



```
[petrovkinal002@fedora lab06]$ nasm -f elf lab6-3.asm
[petrovkinal002@fedora lab06]$ ld -m elf_i386 -o lab6-3 lab6-3.o
[petrovkinal002@fedora lab06]$ ./lab6-3
Результат: 4
Остаток от деления: 1
[petrovkinal002@fedora lab06]$
```

Figure 17: Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения $f(x) = (4 * 6 + 2)/5$ (рис. [18]).



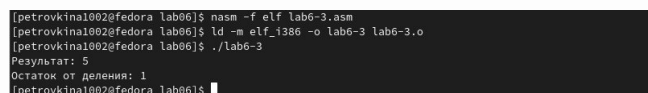
```
lab6-3.asm
~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06

Makefile lab6-1.asm lab

#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 ebx,ebx ; обнуляем EBX для корректной работы div
mov ebx,5 ; EBX=5
div ebx ; EAX=EAX/5, EDI=остаток от деления
mov edi,eax ; запись результата вычисления в "edi"
; ---- Вывод результата на экран
mov eax,div ; вывод подпрограммы печати
call printf ; сообщение "Результат: "
mov eax,edi ; вывод подпрограммы печати значения
call printf ; из "edi" в виде символов
mov eax,rem ; вывод подпрограммы печати значения
call printf ; сообщение "Остаток от деления: "
mov eax,edi ; вывод подпрограммы печати значения
call printf ; из "edi" (остаток) в виде символов
call quit ; вывод подпрограммы завершения
```

Figure 18: Изменение программы

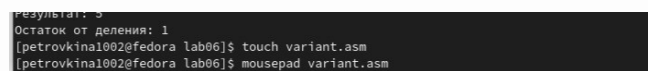
Создаю и запускаю новый исполняемый файл (рис. [19]). Я посчитала для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно.



```
[petrovkinal002@fedora lab06]$ nasm -f elf lab6-3.asm
[petrovkinal002@fedora lab06]$ ld -m elf_i386 -o lab6-3 lab6-3.o
[petrovkinal002@fedora lab06]$ ./lab6-3
Результат: 5
Остаток от деления: 1
[petrovkinal002@fedora lab06]$
```

Figure 19: Запуск исполняемого файла

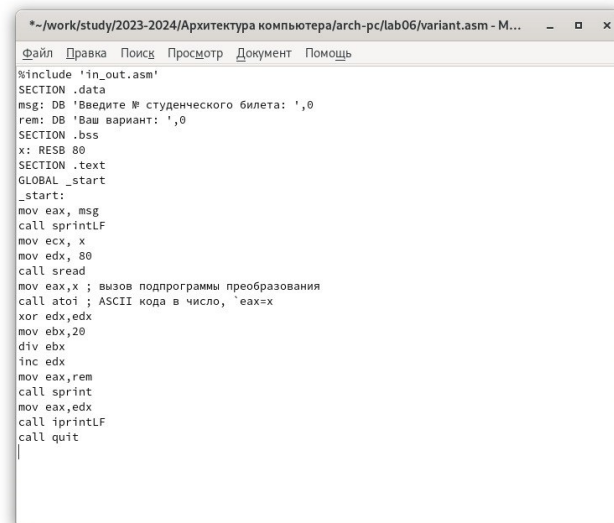
Создаю файл variant.asm с помощью утилиты touch (рис. [20]).



```
Результат: 5
Остаток от деления: 1
[petrovkinal002@fedora lab06]$ touch variant.asm
[petrovkinal002@fedora lab06]$ mousepad variant.asm
```

Figure 20: Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. [21]).

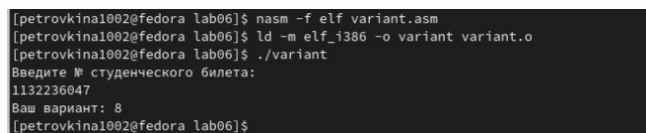


```
*~/work/study/2023-2024/Архитектура компьютера/arch-pc/lab06/variant.asm - M...
Файл  Правка  Поиск  Просмотр  Документ  Помощь

%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 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
inc edx
mov eax,rem
call sprint
mov eax,edx
call iprintf
call quit
|
```

Figure 21: Редактирование файла

Создаю и запускаю исполняемый файл (рис. [22]). Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 8.



```
[petrovkinal002@fedora lab06]$ nasm -f elf variant.asm
[petrovkinal002@fedora lab06]$ ld -m elf_i386 -o variant variant.o
[petrovkinal002@fedora lab06]$ ./variant
Введите № студенческого билета:
1132236047
Ваш вариант: 8
[petrovkinal002@fedora lab06]$
```

Figure 22: Запуск исполняемого файла

4.2.1 Ответы на вопросы по программе

1. За вывод сообщения “Ваш вариант” отвечают строки кода:

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

1. Инструкция `mov ecx, 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

2. `call atoi` используется для вызова подпрограммы из внешнего файла, которая преобразует `ascii`-код символа в целое число и записывает результат в регистр `eax`

3. За вычисления варианта отвечают строки:

```
xor edx,edx ; обнуление edx для корректной работы div
mov ebx,20 ; ebx = 20
div ebx ; eax = eax/20, edx - остаток от деления
inc edx ; edx = edx + 1
```

1. При выполнении инструкции `div ebx` остаток от деления записывается в регистр `edx`

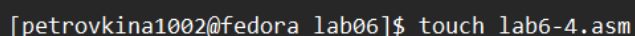
2. Инструкция `inc edx` увеличивает значение регистра `edx` на 1

3. За вывод на экран результатов вычислений отвечают строки:

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

4.3 Выполнение заданий для самостоятельной работы

Создаю файл `lab6-4.asm` с помощью утилиты `touch` (рис. [23]).



```
[petrovkina1002@fedora lab06]$ touch lab6-4.asm
```

Figure 23: Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения $(11 + x) * 2 - 6$ (рис. [24]). Это выражение было под вариантом 8.


```
call sprint ; вызов подпрограммы печати сообщения
mov ecx, x ; запись адреса переменной в ecx
mov edx, 80 ; запись длины вводимого значения в edx
call sread ; вызов подпрограммы ввода сообщения
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, `eax=x`
add eax, 11;  $eax = eax + 11 = x + 11$ 
mov ebx, 2 ; запись значения 2 в регистр ebx
mul ebx;  $EAX = EAX * EBX = (x + 11) * 2$ 
add eax, -6;  $eax = eax - 6 = (x + 11) * 2 - 6$ 
mov edi, eax ; запись результата вычисления в 'edi'
; --- Вывод результата на экран
mov eax, rem ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax, edi ; вызов подпрограммы печати значения
call iprint ; из 'edi' в виде символов
call quit ; вызов подпрограммы завершения
```

5 Выводы

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

6 Список литературы

1. Лабораторная работа №6
2. Таблица ASCII