

Отчет по лабораторной работе №6

Дисциплина: архитектура компьютера

Провоторов Антон Григорьевич

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
4.1	Символьные и численные данные в NASM	9
4.2	Выполнение арифметических операций в NASM	13
4.2.1	Ответы на вопросы по программе	15
4.3	Выполнение заданий для самостоятельной работы	16
5	Выводы	20
6	Список литературы	21

Список иллюстраций

4.1	Создание директории и создание файла	9
4.2	Создание копии файла	9
4.3	Редактирование файла	9
4.4	Запуск исполняемого файла	10
4.5	Редактирование файла	10
4.6	Запуск исполняемого файла	11
4.7	Создание файла	11
4.8	Редактирование файла	11
4.9	Запуск исполняемого файла	11
4.10	Редактирование файла	12
4.11	Запуск исполняемого файла	12
4.12	Редактирование файла	12
4.13	Запуск исполняемого файла	12
4.14	Создание файла	13
4.15	Редактирование файла	13
4.16	Запуск исполняемого файла	13
4.17	Изменение программы	14
4.18	Запуск исполняемого файла	14
4.19	Создание файла	14
4.20	Редактирование файла	15
4.21	Запуск исполняемого файла	15
4.22	Создание файла	16
4.23	Написание программы	17
4.24	Запуск исполняемого файла	17
4.25	Запуск исполняемого файла	17

Список таблиц

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 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.

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. [4.1]). Перехожу в созданный каталог с помощью утилиты `cd`. С помощью утилиты `touch` создаю файл `lab6-1.asm`

```
agprovotorov@dk6n55 ~ $ cd /afs/.dk.sci.pfu.edu.ru/home/a/g/agprovotorov/work/arch-ps
agprovotorov@dk6n55 ~/work/arch-ps $ mkdir lab06
agprovotorov@dk6n55 ~/work/arch-ps $ cd ~/work/arch-ps/lab06
bash: cd: /afs/.dk.sci.pfu.edu.ru/home/a/g/agprovotorov/work/arch-ps/lab06: Нет такого файла или каталога
agprovotorov@dk6n55 ~/work/arch-ps $ cd /afs/.dk.sci.pfu.edu.ru/home/a/g/agprovotorov/work/arch-ps/lab06
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ touch lab6-1.asm
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $
```

Рис. 4.1: Создание директории и создание файла

Копирую в текущий каталог файл `in_out.asm` с помощью утилиты `cp`, т.к. он будет использоваться в других программах (рис. [4.2]).

```
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ cp ~/Зарпужки/in_out.asm in_out.asm
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ ls
in_out.asm  lab6-1.asm
```

Рис. 4.2: Создание копии файла

Открываю созданный файл `lab6-1.asm`, вставляю в него программу вывода значения регистра `eax` (рис. [4.3]).

```
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ nasm -f elf lab6-1.asm
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ ./lab6-1
j
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $
```

Рис. 4.3: Редактирование файла

Создаю исполняемый файл программы и запускаю его (рис. [4.4]). Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.

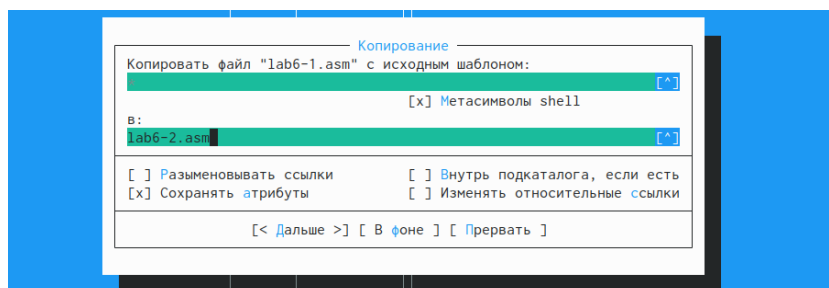


Рис. 4.4: Запуск исполняемого файла

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. [4.5]).

```
%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. 4.5: Редактирование файла

Создаю новый исполняемый файл программы и запускаю его (рис. [4.6]). Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

```
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ nasm -f elf lab6-2.asm
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ ./lab6-2
```

Рис. 4.6: Запуск исполняемого файла

Создаю новый файл lab6-3.asm с помощью утилиты touch (рис. [4.7]).

```
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ touch lab6-3.asm
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $
```

Рис. 4.7: Создание файла

Ввожу в файл текст другой программы для вывода значения регистра eax (рис. [4.8]).

```
GNU nano 6.4 /afs/dk.sci.pfu.edu.ru/home/a/g/agprovotorov/work/arch-ps/lab06/lab6-3.asm
#include "in_out.asm"
SECTION .text
GLOBAL _start
_start:
mov eax, '6'
mov ebx, '4'
add eax, ebx
call iprintf
call quit
```

Рис. 4.8: Редактирование файла

Создаю и запускаю исполняемый файл lab6-3 (рис. [4.9]). 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”.

```
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ nasm -f elf lab6-3.asm
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ ./lab6-3
106
```

Рис. 4.9: Запуск исполняемого файла

Заменяю в тексте программы в файле lab6-3.asm символы “6” и “4” на числа 6 и 4 (рис. [4.10]).

```

lab6-3.asm      [-M--]  9 L:[ 1+ 5  6/ 10] *(77 / 114b) 0010 0x00A
#include "lab6-3.h"
SECTION .text
GLOBAL _start
_start:
mov eax, 6
mov ebx, 4
add eax, ebx
call iprintLF
call quit

```

Рис. 4.10: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. [4.11]). Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

```

agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ nasm -f elf lab6-3.asm
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ ./lab6-3
10

```

Рис. 4.11: Запуск исполняемого файла

Заменяю в тексте программы функцию iprintLF на iprint (рис. [4.12]).

```

lab6-3.asm      [-M--] 11 L:[ 1+ 7  8/ 10] *(101 / 112b) 0010 0x00A
#include "lab6-3.h"
SECTION .text
GLOBAL _start
_start:
mov eax, 6
mov ebx, 4
add eax, ebx
call iprint
call quit

```

Рис. 4.12: Редактирование файла

Создаю и запускаю новый исполняемый файл (рис. [4.13]). 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.

```

agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ nasm -f elf lab6-3.asm
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ ./lab6-3
10agprovotorov@dk6n55 ~/work/arch-ps/lab06 $

```

Рис. 4.13: Запуск исполняемого файла

4.2 Выполнение арифметических операций в NASM

Создаю файл lab6-4.asm с помощью утилиты touch (рис. [4.14]).

```
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ touch lab6-4.asm
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $
```

Рис. 4.14: Создание файла

Ввожу в созданный файл текст программы для вычисления значения выражения $f(x) = (5 * 2 + 3)/3$ (рис. [4.15]).

```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/a/g/agprovotorov/work/
%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: '
mov eax,edi ; вызов подпрограммы печати значения
call iprintf ; из 'edi' в виде символов
mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Остаток от деления: '
mov eax,edx ; вызов подпрограммы печати значения
call iprintf ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения
```

Рис. 4.15: Редактирование файла

Создаю исполняемый файл и запускаю его (рис. [4.16]).

```
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ nasm -f elf lab6-4.asm
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ ld -m elf_i386 -o lab6-4 lab6-4.o
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ ./lab6-4
Результат: 4
Остаток от деления: 1
```

Рис. 4.16: Запуск исполняемого файла

Изменяю программу так, чтобы она вычисляла значение выражения $f(x) = (4 * 6 + 2)/5$ (рис. [4.17]).

```

lab6-4.asm          [-M--]  9 L: [ 1+13 14/ 27] *(423 /1237b) 0032 0x020
#include "lab6-4.h" ; подключение внешнего файла
SECTION .text
div: DB "Результат: ",0
rem: DB "Остаток от деления: ",0
SECTION .data
GLOBAL _start
_start:
; ----- Вычисление выражения:
mov eax,4 ; EAX=5
mov ebx,6 ; EBX=2
mul ebx ; EAX=EAX*EBX
add ebx,2 ; EBX=EAX+2
xor edx,edx ; обнуляем EDI для корректной работы div
mov ebx,5 ; EBX=3
div ebx ; EAX=EAX/2, EDI=остаток от деления
mov edi,eax ; запись результата вычисления в 'edi'
; ----- Вывод результата на экран:
mov eax,div ; вывод подпрограммы печати
call printf ; сообщения "Результат: "
mov eax,edi ; вывод подпрограммы печати значения
call printf ; на "edi" в виде символов
mov eax,rem ; вывод подпрограммы печати
call printf ; сообщения "Остаток от деления: "
mov eax,edi ; вывод подпрограммы печати значения
call printf ; на "edi" (остаток) в виде символов
call quit ; вывод подпрограммы завершения

```

Рис. 4.17: Изменение программы

Создаю и запускаю новый исполняемый файл (рис. [4.18]). Я посчитал для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно.

```

agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ nasm -f elf lab6-4.asm
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ ld -m elf_i386 -o lab6-4 lab6-4.o
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ ./lab6-4
Результат: 5
Остаток от деления: 1

```

Рис. 4.18: Запуск исполняемого файла

Создаю файл variant.asm с помощью утилиты touch (рис. [4.19]).

```

agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ touch variant.asm
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $

```

Рис. 4.19: Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. [4.20]).

```

GNU nano 6.4 /afs/dk.sci.pfu.edu.ru/home/a/g/approvotorov/work/arch-ps/lab06/variant.asm
include 'In_Out.asm'
SECTION .data
msg: DB 'Введите No студенческого билета: ',0
em: DB 'Ваш вариант: ',0
SECTION .bss
resb 80
SECTION .text
GLOBAL _start
_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'
xor edx, edx
mov ebx, 20
div ebx
inc edx
mov eax, rem
call sprint
mov eax, edx
call iprintf
call quit

```

Рис. 4.20: Редактирование файла

Создаю и запускаю исполняемый файл (рис. [4.21]). Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 1.

```

approvotorov@dk6n55 ~/work/arch-ps/lab06 $ nasm -f elf variant.asm
approvotorov@dk6n55 ~/work/arch-ps/lab06 $ ld -m elf_i386 -o variant variant.o
approvotorov@dk6n55 ~/work/arch-ps/lab06 $ ./variant
Введите No студенческого билета:
1122230300
Ваш вариант: 1

```

Рис. 4.21: 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

```

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

4. 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
```

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

6. Инструкция inc edx увеличивает значение регистра edx на 1

7. За вывод на экран результатов вычислений отвечают строки:

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

4.3 Выполнение заданий для самостоятельной работы

Создаю файл lab6-5.asm с помощью Midnight Commander (рис. [4.22]).

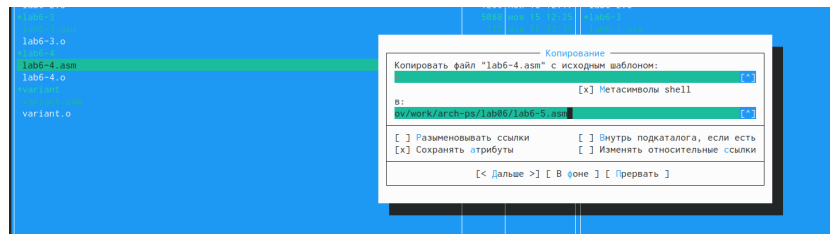


Рис. 4.22: Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения $(10 + x^2) / 3$ (рис. [4.23]). Это выражение было под вариантом 1.


```

GNU nano 6.4 /afs/dk.sci.pfu.edu.ru/home/a/g/agprovotorov/work/arch-ps/lab06/lab6-5.asm
#include "in_out.asm" ; подключение внешнего файла
SECTION .data ; секция инициализированных данных
msg: DB "Введите значение переменной x: ",0
res: DB "Результат: ",0
SECTION .bss ; секция не инициализированных данных
x: RESB 80 ; Переменная, значение x-рой будем вводить с клавиатуры, выделенный размер - 80 байт
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
; ---- Вычисление выражения
mov eax, msg ; запись адреса выводимого сообщения в eax
call printf ; вызов подпрограммы печати сообщения
mov ecx, x ; запись адреса переменной в ecx
mov edx, 80 ; запись длины выводимого значения в edx
call read ; вызов подпрограммы ввода сообщения
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII код в число, eax*х

mov ebx, 2
mul ebx
add eax, 10
xor edx, edx ; обнуляем EDI для корректной работы div
mov ebx, 3 ; 10/3
div ebx

mov edi, eax ; запись результата вычисления в 'edi'

; ---- Вывод результата на экран
mov eax, res ; вызов подпрограммы печати
call printf ; сообщения "Результат: "
mov eax, edi ; вызов подпрограммы печати значения
call printf ; из 'edi' в виде символов
mov eax, res ; вызов подпрограммы печати
call printf ; сообщения "Остаток от деления: "
mov eax, edx ; вызов подпрограммы печати значения
call printf ; из 'edx' (остаток) в виде символов

```

Рис. 4.23: Написание программы

Создаю и запускаю исполняемый файл (рис. [4.24]). При вводе значения 1, вывод - 4.

```

agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ nasm -f elf lab6-5.asm
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ nasm -f elf lab6-5.asm
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ ld -m elf_i386 -o lab6-5 lab6-5.o
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ ./lab6-5
Введите значение переменной x: 1
Результат: 4
Результат: 0

```

Рис. 4.24: Запуск исполняемого файла

Провожу еще один запуск исполняемого файла для проверки работы программы с другим значением на входе (рис. [4.25]). Программа отработала верно.

```

agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ nasm -f elf lab6-5.asm
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ ld -m elf_i386 -o lab6-5 lab6-5.o
agprovotorov@dk6n55 ~/work/arch-ps/lab06 $ ./lab6-5
Введите значение переменной x: 10
Результат: 10
Результат: 0

```

Рис. 4.25: Запуск исполняемого файла

****Листинг 4.1. Программа для вычисления значения выражения $(10 + x*2) / 3$.**

```

#include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; секция инициализированных данных

```

```

msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0
SECTION .bss ; секция не инициированных данных
x: RESB 80 ; Переменная, значение к-рой будем вводить с клавиатуры, выделенный ра
SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу
; ---- Вычисление выражения
mov eax, msg ; запись адреса выводимого сообщения в eax
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

mov ebx,2
mul ebx
add eax,10
xor edx,edx ; обнуляем EDX для корректной работы div
mov ebx,3 ; EBX=3
div ebx

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 iprintLF ; из 'edi' в виде символов
mov eax,rem ; вызов подпрограммы печати
call sprint ; сообщения 'Остаток от деления: '
mov eax,edx ; вызов подпрограммы печати значения
call iprintLF ; из 'edx' (остаток) в виде символов
call quit ; вызов подпрограммы завершения
```

5 Выводы

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

6 Список литературы

1. Лабораторная работа №7
2. Таблица ASCII