

Отчёт по лабораторной работе №6

Дисциплина: архитектура компьютера

Кузнецова Елизавета Андреевна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	9
5	Ответы на вопросы по программе	16
6	Выполнение заданий для самостоятельной работы	17
7	Программа для вычисления значения выражения $x^{3*1/3}+21$	19
8	Выводы	21

Список иллюстраций

4.1	Создание директории	9
4.2	Создание файла	9
4.3	Копирование файла	9
4.4	Редактирование файла	10
4.5	Запуск исполняемого файла	10
4.6	Редактирование файла	10
4.7	Создание и запуск нового исполняемого файла	11
4.8	Создание файла	11
4.9	Редактирование файла	11
4.10	Запуск исполняемого файла	11
4.11	Редактирование файла	12
4.12	Запуск исполняемого файла	12
4.13	Редактирование файла	12
4.14	Запуск исполняемого файла	12
4.15	Создание файла	13
4.16	Редактирование файла	13
4.17	Запуск исполняемого файла	13
4.18	Редактирование программы	14
4.19	Запуск исполняемого файла	14
4.20	Создание файла	14
4.21	Редактирование файла	15
4.22	Запуск исполняемого файла и вывод номера моего варианта . . .	15
6.1	Создание файла	17
6.2	Написание и редактирование программы	17
6.3	Запуск исполняемого файла	18

Список таблиц

1 Цель работы

Цель лабораторной работы - освоение арифметических инструкций языка ассемблера NASM.

2 Задание

1. Освоить символьные и численные данные в NASM.
2. Выполнить арифметические операции в NASM.
3. Написать программу вычисления выражения $x = x(x)$. Вариант узнать по программе для студенческого билета.
4. Загрузить файлы на Github.

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. 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. 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. Расширенная таблица ASCII состоит из двух частей. Первая (символы с кодами 0-127) является универсальной, а вторая (коды 128-255) предназначена для специальных символов и букв национальных алфавитов и на компьютерах разных типов может меняться. 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 делает 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

С помощью утилиты `mkdir` создала директорию, в которой буду создавать файлы с программами для лабораторной работы. Перешла в созданный каталог с помощью утилиты `cd` (рис. [4.1]).

```
eakuznecova@dk8n68 ~ $ mkdir ~/work/arch-pc/lab06
eakuznecova@dk8n68 ~ $ cd ~/work/arch-pc/lab06
```

Рис. 4.1: Создание директории

С помощью утилиты `touch` создала файл `lab6-1.asm` (рис. [4.2]).

```
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $ touch lab6-1.asm
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $ ls
lab6-1.asm
```

Рис. 4.2: Создание файла

Скопировала в текущий каталог файл `in_out.asm` с помощью утилиты `cp`, так как он будет использоваться в других программах (рис. [4.3]).

```
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $ cp ~/Зарядки/in_out.asm in_out.asm
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $ ls
in_out.asm lab6-1.asm
```

Рис. 4.3: Копирование файла

Открыла созданный файл `lab6-1.asm`, вставила в него программу вывода значения регистра `eax` (рис. [4.4]).

```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab06/lab6-1.asm
#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    sprintf
call    quit
```

Рис. 4.4: Редактирование файла

Создала исполняемый файл программы и запустила его. Вывелся символ j, потому что программа вывела символ, соответствующий по система ASCII сумме двоичных кодов и символов 4 и 6 (рис. [4.5]).

```
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $ ./lab6-1
j
```

Рис. 4.5: Запуск исполняемого файла

Изменила в тексте программы символы “6” и “4” на цифры 6 и 4 (рис. [4.6]).

```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab06/lab6-1.asm
#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    sprintf
call    quit
```

Рис. 4.6: Редактирование файла

Создала новый исполняемый файл программы и запустила его. Теперь вывелся символ с кодом 10, это символ перевода строки, он не отображается при выводе на экран (рис. [4.7]).

```
eakuznecova@dk8n68 ~ $ cd ~/work/arch-pc/lab06
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $ nasm -f elf lab6-1.asm
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-1 lab6-1.o
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $ ./lab6-1
```

Рис. 4.7: Создание и запуск нового исполняемого файла

Создала новый файл lab6-2.asm с помощью утилиты touch (рис. [4.8]).

```
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $ touch lab6-2.asm
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $
```

Рис. 4.8: Создание файла

Ввела в файл текст другой программы для вывода значения регистра eax (рис. [4.9]).

```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab06/lab6-2.asm
#include "in_out.asm"

SECTION .text
GLOBAL _start
_start:

mov eax,'6'
mov ebx,'4'
add eax,ebx
call iprintLF

call quit
```

Рис. 4.9: Редактирование файла

Создала исполняемый файл lab6-2.asm и запустила его. Теперь вывелось число 106, потому что программа позволяет вывести именно число, а не символ, но все равно происходит именно сложение кодов символов “6” и “4”(рис. [4.10]).

```
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $ ./lab6-2
106
```

Рис. 4.10: Запуск исполняемого файла

Заменяла в тексте программы в файле lab6-2.asm символы “6” и “4” на цифры 6 и 4 (рис. [4.11]).

```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab06/lab6-2.asm
#include "in_out.asm"

SECTION .text
GLOBAL _start
_start:

mov     eax,6
mov     ebx,4
add     eax,ebx
call    iprintLF

call    quit
```

Рис. 4.11: Редактирование файла

Создала новый исполняемый и запустила его. Теперь программа сложила не соответствующие символам коды в системе ASCII, а цифры, поэтому вывод 10 (рис. [4.12]).

```
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $ ./lab6-2
10
```

Рис. 4.12: Запуск исполняемого файла

Заменяла в тексте программы iprintLF на iprint (рис. [4.13]).

```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab06/lab6-2.asm
#include "in_out.asm"

SECTION .text
GLOBAL _start
_start:

mov     eax,6
mov     ebx,4
add     eax,ebx
call    iprint

call    quit
```

Рис. 4.13: Редактирование файла

Создала новый исполняемый файл и запустила его. Выводилась также сумма цифр 6 и 4, 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 (рис. [4.14]).

```
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $ nasm -f elf lab6-2.asm
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-2 lab6-2.o
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $ ./lab6-2
10eakuznecova@dk8n68 ~/work/arch-pc/lab06 $
```

Рис. 4.14: Запуск исполняемого файла

Создала файл lab6-3.asm с помощью утилиты touch (рис. [4.15]).

```
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $ touch lab6-3.asm
```

Рис. 4.15: Создание файла

Ввела в созданный файл текст программы для вычисления значения выражения $f(x) = (5 * 2 + 3)/3$ (рис. [4.16]).

```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab06/lab6-3.asm
;-----
; Программа вычисления выражения
;-----

%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 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
```

Рис. 4.16: Редактирование файла

Создала исполняемый файл и запустила его (рис. [4.17]).

```
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 4
Остаток от деления: 1
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $
```

Рис. 4.17: Запуск исполняемого файла

Изменила программу, чтобы она вычисляла значение выражения $f(x) = (4 * 6 + 2)/5$ (рис. [4.18]).

```

GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab06/lab6-3.asm
; Программа вычисления выражения
;-----
%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 edx,edx    ; обнуляем EDX для корректной работы div
mov ebx,5      ; EBX=5
div ebx        ; EAX=EAX/5, 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 iprintfLF ; 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 iprintfLF ; из 'edx' (остаток) в виде символов

call quit      ; вызов подпрограммы завершения

```

Рис. 4.18: Редактирование программы

Создала новый исполняемый файл и запустила его. Программа правильно была выполнена, я ее перепроверяла самостоятельно (рис. [4.19]).

```

eakuznecova@dk8n68 ~/work/arch-pc/lab06 $ nasm -f elf lab6-3.asm
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-3 lab6-3.o
eakuznecova@dk8n68 ~/work/arch-pc/lab06 $ ./lab6-3
Результат: 5
Остаток от деления: 1

```

Рис. 4.19: Запуск исполняемого файла

Создала файл variant.asm с помощью утилиты touch (рис. [4.20]).

```

eakuznecova@dk8n52 ~/work/arch-pc/lab06 $ touch variant.asm

```

Рис. 4.20: Создание файла

Ввела в созданный файл текст программы для вычисления варианта задания по номеру студенческого билета (рис. [4.21]).

```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab06/variant.asm
;-----
; Программа вычисления варианта
;-----
%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
call atoi      ; вызов подпрограммы преобразования
               ; ASCII кода в число, 'eax=x'

xor edx,edx
mov ebx,20
div ebx
inc edx

mov eax,rem
call sprintf
mov eax,edx
call iprintf
call quit
```

Рис. 4.21: Редактирование файла

Создала новый исполняемый файл и запустила его. Ввела номер своего студенческого билета с клавиатуры, программа вывела, что мой вариант-20 (рис. [4.22]).

```
eakuznecova@dk8n52 ~/work/arch-pc/lab06 $ nasm -f elf variant.asm
eakuznecova@dk8n52 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o variant variant.o
eakuznecova@dk8n52 ~/work/arch-pc/lab06 $ ./variant
Введите № студенческого билета:
1132236119
Ваш вариант: 20
```

Рис. 4.22: Запуск исполняемого файла и вывод номера моего варианта

5 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  
mov ebx, 20  
div ebx  
inc edx
```

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
```


6 Выполнение заданий для самостоятельной работы

Создала файл lab6-4.asm с помощью утилиты touch (рис. [6.1]).

```
eakuznecova@dk8n52 ~/work/arch-pc/lab06 $ touch lab6-4.asm
```

Рис. 6.1: Создание файла

Открыла созданный файл и ввела в него текст программы для вычисления значения выражения $x^3 \cdot 1/3 + 21$, которое было под 20 вариантом (рис. [6.2]).

```
GNU nano 6.4 /afs/.dk.sci.pfu.edu.ru/home/e/a/eakuznecova/work/arch-pc/lab06/lab6-4.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 которой 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 sprint ; вызов подпрограммы печати сообщения
mov ecx, x ; запись адреса переменной в ecx
mov edx, 80 ; запись длины вводного значения в edx
call sread ; вызов подпрограммы ввода сообщения
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII код в число, 'eax=x'
mov ebx, eax ; записать значение eax в регистр ebx
mul ebx, ebx ; EAX=EAX*EBX
mul ebx, ebx ; EAX=EAX*EBX
mov ebx,1 ; записать значение 1 в регистр ebx
mul ebx, ebx ; EAX=EAX*EBX
mov ebx,3 ; записать значение 3 в регистр ebx
div ebx, ebx ; EAX=EAX/3
add eax,21 ; eax = eax*21 = x*x*x*1/3+21
mov edi,eax ; запись результата вычисления в 'edi'

; ---- Вывод результата на экран
mov eax, res ; вызов подпрограммы печати
call sprint ; сообщения 'Результат: '
mov eax, edi ; вызов подпрограммы печати значения
call iprintf ; из 'edi' в виде символов
call quit ; вызов подпрограммы завершения
```

Рис. 6.2: Написание и редактирование программы

Создала новый исполняемый файл и запустила его. При вводе значения 1, выводом является число 21. При вводе значения 3, выводом является число 30.

Программа сработала верно, я проверяла самостоятельно (рис. [6.3]).

```
eakuznecova@dk8n52 ~/work/arch-pc/lab06 $ nasm -f elf lab6-4.asm
eakuznecova@dk8n52 ~/work/arch-pc/lab06 $ ld -m elf_i386 -o lab6-4 lab6-4.o
eakuznecova@dk8n52 ~/work/arch-pc/lab06 $ ./lab6-4
Введите значение переменной x: 1
Результат: 21
eakuznecova@dk8n52 ~/work/arch-pc/lab06 $ ./lab6-4
Введите значение переменной x: 3
Результат: 30
```

Рис. 6.3: Запуск исполняемого файла

7 Программа для вычисления значения выражения $x^3 \cdot \frac{1}{3} + 21$

```
%include 'in_out.asm' ; подключение внешнего файла
SECTION .data ; секция инициированных данных
msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0
SECTION .bss ; секция не инициированных данных
x: RESB 80 ; Переменная, значение которой будем вводить с клавиатуры, выделенный
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, eax ; записать значение eax в регистр ebx
mul ebx ; EAX=EAX*EBX
```

```

mul ebx; EAX=EAX*EBX
mov ebx,1; записать значение 1 в регистр ebx
mul ebx; EAX=EAX*EBX
mov ebx,3; записать значение 3 в регистр ebx
div ebx; EAX=EAX/3
add eax,21; eax = eax+21 = x*x*x*1/3+21
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
call quit ; вызов подпрограммы завершения

```

8 Выводы

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