

Отчёт по лабораторной работе №6

Дисциплина: Архитектура компьютера

Абрамова Ульяна Михайловна

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
4.3	Ответы на вопросы по программе	16
4.4	Выполнение заданий для самостоятельной работы	17
5	Выводы	20
6	Список литературы	21

Список иллюстраций

4.1	Создание файлов	9
4.2	Написание программы	9
4.3	Запуск программы	10
4.4	Изменение текста программы	10
4.5	Создание файла программы и его запуск	10
4.6	Написание программы в новом созданном файле	11
4.7	Создание и запуск файла программы	11
4.8	Изменение текста программы	12
4.9	Создание и запуск файла программы	12
4.10	Изменение текста программы	13
4.11	Создание и запуск файла программы	13
4.12	Написание программы в новом созданном файле	14
4.13	Создание и запуск файла программы	14
4.14	Изменение текста программы	15
4.15	Создание и запуск файла программы	15
4.16	Написание программы в новом созданном файле	16
4.17	Создание и запуск файла программы	16
4.18	Написание программы	18
4.19	Создание, запуск и проверка корректности программы	18

List of Tables

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. Перехожу в созданный каталог с помощью утилиты `cd`. Благодаря утилите `touch` создаю файл `lab6-1.asm` (рис. 4.1).

```
uliana_abramova@fedora:~$ mkdir ~/work/arch-pc/lab06
uliana_abramova@fedora:~$ cd ~/work/arch-pc/lab06
uliana_abramova@fedora:~/work/arch-pc/lab06$ touch lab6-1.asm
```

Рис. 4.1: Создание файлов

Открываю созданный файл `lab6-1.asm`, вставляю в него программу вывода значения регистра `eax` (рис. 4.2).

```
GNU nano 7.2 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. 4.2: Написание программы

Создаю исполняемый файл программы и запускаю его (рис. 4.3). Вывод программы: символ j, потому что программа вывела символ, соответствующий по системе ASCII сумме двоичных кодов символов 4 и 6.

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

Рис. 4.3: Запуск программы

Изменяю в тексте программы символы '6' и '4' на цифры 6 и 4 (рис. 4.4).

```
GNU nano 7.2 lab6-1
#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
```

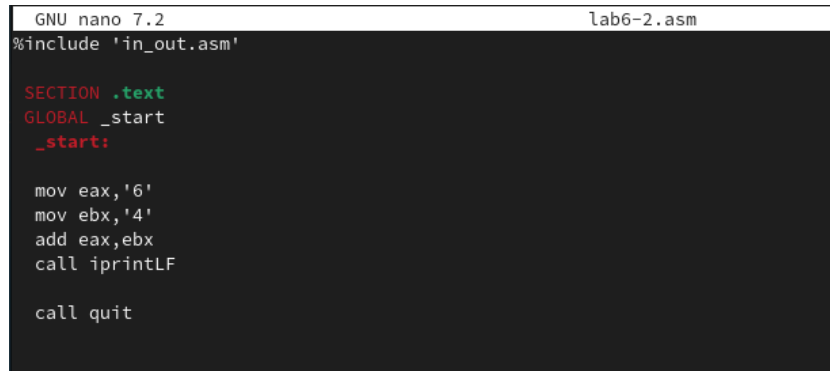
Рис. 4.4: Изменение текста программы

Создаю новый исполняемый файл программы и запускаю его (рис. 4.5). Теперь вывелся символ с кодом 10, это символ перевода строки, этот символ не отображается при выводе на экран.

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

Рис. 4.5: Создание файла программы и его запуск

Создаю новый файл lab6-2.asm с помощью утилиты touch и ввожу в него текст другой программы для вывода значения регистра eax (рис. 4.6).



```
GNU nano 7.2 lab6-2.asm
#include 'in_out.asm'

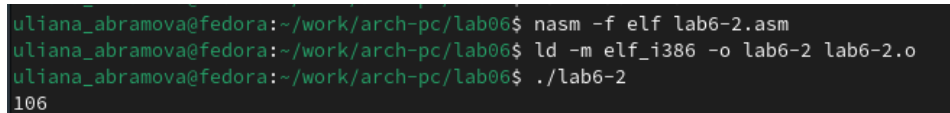
SECTION .text
GLOBAL _start
_start:

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

call quit
```

Рис. 4.6: Написание программы в новом созданном файле

Создаю и запускаю исполняемый файл lab6-2. Теперь выводится число 106, потому что программа позволяет вывести именно число, а не символ, хотя все еще происходит сложение кодов символов '6' и '4'. (рис. 4.7).



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

Рис. 4.7: Создание и запуск файла программы

Заменяю в тексте программы в файле lab6-2.asm символы '6' и '4' на числа 6 и 4 (рис. 4.8).

```
GNU nano 7.2 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. 4.8: Изменение текста программы

Создаю и запускаю новый исполняемый файл. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому выводится 10 (рис. 4.9).

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

Рис. 4.9: Создание и запуск файла программы

Заменяю в тексте программы функцию `iprintLF` на `iprint` (рис. 4.10).

```
GNU nano 7.2 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. 4.10: Изменение текста программы

Создаю и запускаю новый исполняемый файл. 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` (рис. 4.11).

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

Рис. 4.11: Создание и запуск файла программы

4.2 Выполнение арифметических операций в NASM

Создаю файл `lab6-3.asm` с помощью утилиты `touch` и ввожу в него текст программы для вычисления значения выражения $f(x) = (5 * 2 + 3)/3$ (рис. 4.12).

```
GNU nano 7.2                                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
```

Рис. 4.12: Написание программы в новом созданном файле

Создаю исполняемый файл и запускаю его (рис. 4.13).

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

Рис. 4.13: Создание и запуск файла программы

Изменяю программу так, чтобы она вычисляла значение выражения $f(x) = (4 * 6 + 2)/5$ (рис. 4.14).

```
GNU nano 7.2 lab6-3.asm
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
```

Рис. 4.14: Изменение текста программы

Создаю и запускаю новый исполняемый файл. Я посчитала для проверки правильности работы программы значение выражения самостоятельно, программа отработала верно. (рис. 4.15)

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

Рис. 4.15: Создание и запуск файла программы

Создаю файл variant.asm с помощью утилиты touch и ввожу в него текст программы для вычисления варианта задания по номеру студенческого билета (рис. 4.16).

```

GNU nano 7.2                                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 sprintLF

```

Рис. 4.16: Написание программы в новом созданном файле

Создаю и запускаю исполняемый файл. Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 3 (рис. 4.17).

```

uliana_abramova@fedora:~/work/arch-pc/lab06$ nasm -f elf variant.asm
uliana_abramova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
uliana_abramova@fedora:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1132246782
Ваш вариант: 3

```

Рис. 4.17: Создание и запуск файла программы

4.3 Ответы на вопросы по программе

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

```

mov eax,rem
call sprint

```

2. Инструкция `mov esx, x` используется, чтобы положить адрес вводимой строки `x` в регистр `esx` `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
```

4.4 Выполнение заданий для самостоятельной работы

Создаю файл `lab7-4.asm` с помощью утилиты `touch`, открываю его для редактирования, ввожу в него текст программы для вычисления значения выражения $(x+2)^2$, которое было под вариантом 3 (рис. 4.18).

```

GNU nano 7.2                                v3.asm
#include 'in_out.asm' ; подключение внешнего файла
SECTION .data
msg: DB 'Введите значение переменной x: ',0
rem: DB 'Результат: ',0
SECTION .bss
x: RESB 80 ; Переменная, значение которой будем вводить с клавиатуры
SECTION .text
GLOBAL _start
_start:
; ---- Вычисление выражения
mov eax, msg
call sprint
mov ecx, x
mov edx, 80
call sread
mov eax, x ; вызов подпрограммы преобразования
call atoi ; ASCII кода в число, 'eax=x'
add eax, 2; eax = eax+2 = x + 2
mov ebx, eax
mul ebx; EAX=EAX*EBX = (x+2)^2

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 iprintfLF ; из 'edi' в виде символов
call quit ; вызов подпрограммы завершения

```

Рис. 4.18: Написание программы

Создаю и запускаю исполняемый файл. При вводе значения на входе 2, вывод - 16. Провожу еще один запуск исполняемого файла для проверки работы программы, но с другим значением равным 8, вывод - 100. Что показывает верную отработку программы (рис. 4.19).

```

uliana_abramova@fedora:~/work/arch-pc/lab06$ nasm -f elf v3.asm
uliana_abramova@fedora:~/work/arch-pc/lab06$ ld -m elf_i386 -o v3 v3.o
uliana_abramova@fedora:~/work/arch-pc/lab06$ ./v3
Введите значение переменной x: 2
Результат: 16
uliana_abramova@fedora:~/work/arch-pc/lab06$ ./v3
Введите значение переменной x: 8
Результат: 100

```

Рис. 4.19: Создание, запуск и проверка корректности программы

Программа для вычисления значения выражения $(x+2)^2$:

```

#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 к-рой будем вводить с клавиатуры, выделенный размер
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`
add eax,2; eax = eax+2 = x + 2
mov ebx,eax ; запись значения eax в регистр ebx
mul ebx; EAX=EAX*EBX = (x+2)^2
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. Архитектура ЭВМ
2. Таблица ASCII