

Отчет по лабораторной работе №6

Дисциплина архитектура компьютера

Ахатов Эмиль Эрнстович

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	6
4	Выполнение лабораторной работы	8
4.1	Символьные и численные данные в NASM	8
4.2	Выполнение арифметических операций в NASM	11
4.3	Ответы на вопросы по программе	15
5	Выводы	17

Список иллюстраций

4.1	Создание директории	8
4.2	Создание файла	8
4.3	Ввод программы	9
4.4	Запуск исполняемого файла	9
4.5	Редактирование файла	9
4.6	Запуск исполняемого файла	9
4.7	Создание файла	10
4.8	Редактирование файла	10
4.9	Запуск исполняемого файла	10
4.10	Редактирование файла	10
4.11	Запуск исполняемого файла	11
4.12	Редактирование файла	11
4.13	Запуск исполняемого файла	11
4.14	Создание файла	11
4.15	Редактирование файла	12
4.16	Запуск исполняемого файла	12
4.17	Редактирование файла	13
4.18	Запуск исполняемого файла	13
4.19	Создание файла	13
4.20	Редактирование файла	14
4.21	Запуск исполняемого файла	14
4.22	Создание файла	16
4.23	Редактирование файла	16

1 Цель 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, перехожу в директорию.

```
emil@fedora:~/study_2024-2025_arhpc/lab06$ mkdir lab06
emil@fedora:~/study_2024-2025_arhpc/lab06$ cd lab06
emil@fedora:~/local/share/Trash/files/lab06/lab06$
```

Рис. 4.1: Создание директории

С помощью утилиты `touch` создаю файл `lab6-1.asm`

```
emil@fedora:~/study_2024-2025_arhpc/lab06$ touch lab6-1.asm
emil@fedora:~/study_2024-2025_arhpc/lab06$ ls
lab6-1.asm
emil@fedora:~/study_2024-2025_arhpc/lab06$
```

Рис. 4.2: Создание файла

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


```

#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. 4.3: Ввод программы

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

```

emil@fedora:~/study_2024-2025_arhpc/lab06$ nasm -f elf lab6-1.asm
emil@fedora:~/study_2024-2025_arhpc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
emil@fedora:~/study_2024-2025_arhpc/lab06$ ./lab6-1
j

```

Рис. 4.4: Запуск исполняемого файла

Изменяю в тексте программы символы “6” и “4” на цифры 6 и 4

```

#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. 4.5: Редактирование файла

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

```

emil@fedora:~/study_2024-2025_arhpc/lab06$ nasm -f elf lab6-1.asm
emil@fedora:~/study_2024-2025_arhpc/lab06$ ld -m elf_i386 -o lab6-1 lab6-1.o
emil@fedora:~/study_2024-2025_arhpc/lab06$ ./lab6-1

```

Рис. 4.6: Запуск исполняемого файла

Создаю новый файл lab6-2.asm с помощью утилиты touch.

```
emil@fedora:~/study_2024-2025_arhpc/lab06$ touch lab6-2.asm
emil@fedora:~/study_2024-2025_arhpc/lab06$ ls
in_out.asm  lab6-1  lab6-1.asm  lab6-1.o  lab6-2.asm
emil@fedora:~/study_2024-2025_arhpc/lab06$
```

Рис. 4.7: Создание файла

Ввожу в файл текст другой программы для вывода значения регистра eax

```
%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: Редактирование файла

Создаю и запускаю исполняемый файл lab6-2. Теперь вывод числа 106, программа позволяет вывести именно число, а не символ, хотя все еще происходит именно сложение кодов символов “6” и “4”.

```
emil@fedora:~/study_2024-2025_arhpc/lab06$ nasm -f elf lab6-2.asm
emil@fedora:~/study_2024-2025_arhpc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
emil@fedora:~/study_2024-2025_arhpc/lab06$ ./lab6-2
106
emil@fedora:~/study_2024-2025_arhpc/lab06$
```

Рис. 4.9: Запуск исполняемого файла

Заменяю в тексте программы в файле lab6-2.asm символы “6” и “4” на числа 6 и 4

```
%include 'in_out.asm'
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. Теперь программа складывает не соответствующие символам коды в системе ASCII, а сами числа, поэтому вывод 10.

```
emil@fedora:~/study_2024-2025_arhpc/lab06$ nasm -f elf lab6-2.asm
emil@fedora:~/study_2024-2025_arhpc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
emil@fedora:~/study_2024-2025_arhpc/lab06$ ./lab6-2
106
```

Рис. 4.11: Запуск исполняемого файла

Заменяю в тексте программы функцию `iprintLF` на `iprint`

```
%include 'in_out.asm'
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. 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`.

```
emil@fedora:~/study_2024-2025_arhpc/lab06$ nasm -f elf lab6-2.asm
emil@fedora:~/study_2024-2025_arhpc/lab06$ ld -m elf_i386 -o lab6-2 lab6-2.o
emil@fedora:~/study_2024-2025_arhpc/lab06$ ./lab6-2
10emil@fedora:~/study_2024-2025_arhpc/lab06$
```

Рис. 4.13: Запуск исполняемого файла

4.2 Выполнение арифметических операций в NASM

Создаю файл `lab6-3.asm` с помощью утилиты `touch`

```
10emil@fedora:~/study_2024-2025_arhpc/lab06$ touch lab6-3.asm
emil@fedora:~/study_2024-2025_arhpc/lab06$ ls
in_out.asm  lab6-1.asm  lab6-2      lab6-2.o
lab6-1      lab6-1.o    lab6-2.asm  lab6-3.asm
emil@fedora:~/study_2024-2025_arhpc/lab06$
```

Рис. 4.14: Создание файла

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

```
%include 'in_quit.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;
mov ebx,2;
mul ebx;
add eax,3;
xor edx,edx;
mov ebx,3;
div ebx;
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

```
emil@fedora:~/study_2024-2025_arhpc/lab06$ nasm -f elf lab6-3.asm
emil@fedora:~/study_2024-2025_arhpc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
emil@fedora:~/study_2024-2025_arhpc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
```

Рис. 4.16: Запуск исполняемого файла

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

```

%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 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. 4.17: Редактирование файла

Создаю и запускаю новый исполняемый файл, программа выполняется верно

```

emil@fedora:~/study_2024-2025_arhpc/lab06$ nasm -f elf lab6-3.asm
emil@fedora:~/study_2024-2025_arhpc/lab06$ ld -m elf_i386 -o lab6-3 lab6-3.o
emil@fedora:~/study_2024-2025_arhpc/lab06$ ./lab6-3
Результат: 5
Остаток от деления: 1
emil@fedora:~/study_2024-2025_arhpc/lab06$ 

```

Рис. 4.18: Запуск исполняемого файла

Создаю файл variant.asm с помощью утилиты touch.

```

emil@fedora:~/study_2024-2025_arhpc/lab06$ touch variant.asm
emil@fedora:~/study_2024-2025_arhpc/lab06$ ls
in_out.asm  lab6-1.asm  lab6-2      lab6-2.o  lab6-3.asm  variant.asm
lab6-1      lab6-1.o   lab6-2.asm  lab6-3    lab6-3.o
emil@fedora:~/study_2024-2025_arhpc/lab06$ 

```

Рис. 4.19: Создание файла

Ввожу в файл текст программы для вычисления варианта задания по номеру студенческого билета

```

%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
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 iprintLF

```

Рис. 4.20: Редактирование файла

Создаю и запускаю исполняемый файл. Ввожу номер своего студ. билета с клавиатуры, программа вывела, что мой вариант - 12

```

emil@fedora:~/study_2024-2025_arhpc/lab06$ nasm -f elf variant.asm
emil@fedora:~/study_2024-2025_arhpc/lab06$ ld -m elf_i386 -o variant variant.o
emil@fedora:~/study_2024-2025_arhpc/lab06$ ./variant
Введите № студенческого билета:
1132242471
Ваш вариант: 12

```

Рис. 4.21: Запуск исполняемого файла

4.3 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
```

#Выполнение заданий для самостоятельной работы

Создаю файл lab6-4.asm с помощью утилиты touch

in_out.asm	lab6-1.o	lab6-2.o	lab6-3.o	variant.asm
lab6-1	lab6-2	lab6-3	lab6-4.asm	variant.o
lab6-1.asm	lab6-2.asm	lab6-3.asm	variant	

Рис. 4.22: Создание файла

Открываю созданный файл для редактирования, ввожу в него текст программы для вычисления значения выражения $(8x - 6)/2$

```
%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 неинициализированных данных
x: RESB 80 ; Переменная для хранения значения, вводимого с клавиатуры

SECTION .text ; Код программы
GLOBAL _start ; Начало программы
_start: ; Точка входа в программу

; ---- Ввод значения x ----
mov eax, msg ; Адрес выводимого сообщения в eax
call sprintf ; Вызов подпрограммы для печати сообщения
mov ecx, x ; Запись адреса переменной x в ecx
mov edx, 80 ; Указание длины выводимого значения в edx
call read ; Вызов подпрограммы для ввода значения
mov eax, x ; Преобразование ASCII в число
call atoi ; Преобразование ASCII кода в число, теперь 'eax = x'

; ---- Вычисление выражения (8x - 6) / 2 ----
mov ebx, 8 ; Установка значения 8 в ebx
mul ebx ; Умножение: EAX = EAX * EBX = x * 8
sub eax, 6 ; Вычитание 6: EAX = EAX - 6 = 8x - 6
mov ebx, 2 ; Установка значения 2 в ebx для деления
cdq ; Расширение EAX в EDI:EAX для корректной работы div
```

Рис. 4.23: Редактирование файла

Создаю и запускаю исполняемый файл, при вводе значения 1, вывод 1. При вводе значения 5, вывод 17. Программа отработала верно.

5 Выводы

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