РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ
ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

дисциплина: Архитектура компьютера

Студент: Юсупова Алина Руслановна

Группа: НКАбд-06-25

МОСКВА

2025 г.

Содержание

1 Цель работы

Целью работы является освоение арифметических инструкций языка ассемблера NASM.

2 Задания

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

Создаю каталог для программ лабораторной работы №6 и перехожу в него, создаю там файл lab6-1.asm(рис. 1).

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

Рис. 1. Создание нового каталога и файла lab6-1.asm.

В созданном файле ввожу программу из листинга (рис. 2).

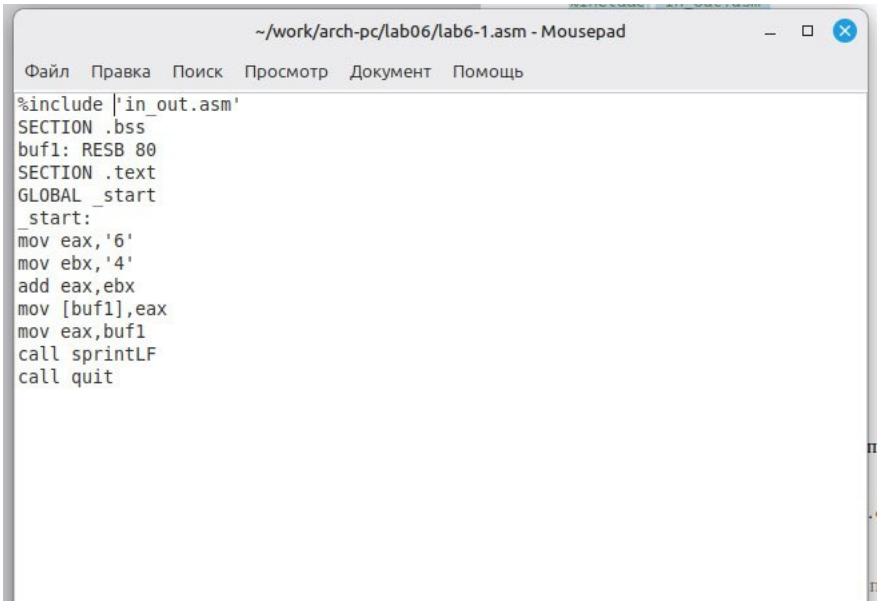

Рис. 2. Сохранение новой программы.

Создаю исполняемый файл и запускаю его, вывод программы отличается от предполагаемого изначально, ибо коды символов в сумме дают символ j по таблице ASCII. {#fig:003 width=70%}

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

Рис. 3. Запуск изначальной программы.

Изменяю текст изначальной программы, убрав кавычки в 7 и 8 строках (рис. 4).

Рис. 4. Измененная программа lab6-1.asm.

На этот раз программа выдала пустую строчку, это связано с тем, что символ 10 означает переход на новую строку (рис. 5).

Рис. 5. Запуск измененной программы lab6-1.asm.

Создаю новый файл lab6-2.asm для будущей программы и проверяю его наличие в каталоге lab06 (рис. 6).

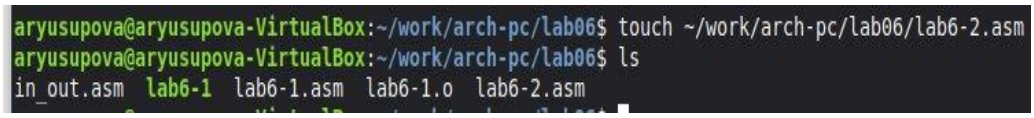

Рис. 6. Создание файла lab6-2.asm.

Далее вставляю код с листинга в файл lab6-2.asm (рис.7).

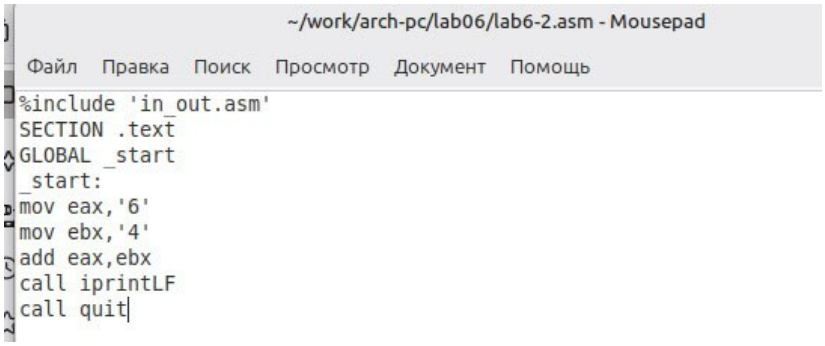

Рис.7. Программа lab6-2.asm.

Затем запускаю исполняемый файл lab6-2.asm. Теперь отображается результат 106, программа, как и в первый раз, сложила коды символов, но вывела само число, а не его символ, благодаря замене функции вывода на iprintLF (рис. 8).

Рис. 8. Запуск неизменённой программы lab6-2.asm.

Убрав кавычки в программе, я снова ее запускаю и получаю предполагаемый изначально результат (рис. 9).

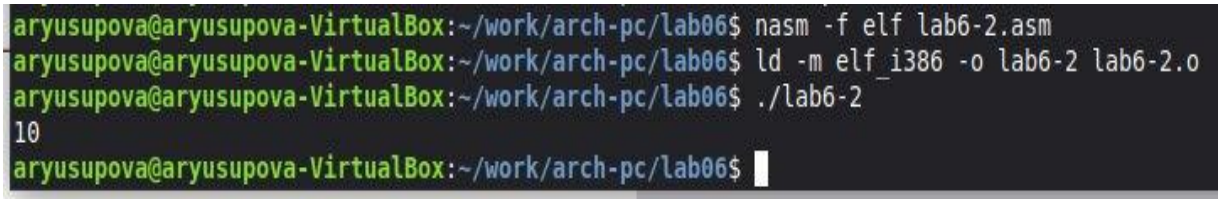

Рис. 9. Запуск измененной программы lab6-2.asm.

Заменив функцию вывода на iprint, я получаю тот же результат, но без переноса строки (рис. 10).

Рис.10. Замена функции вывода в программе lab6-2.asm.

4.2 Выполнение арифметических операций в NASM

Создаю новый файл lab6-3.asm, копирую и вставляю в него содержимое листинга (рис.11-12).

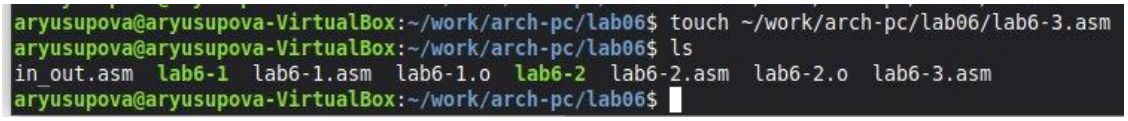

Рис.11. Создание файла lab6-3.asm.

~/work/arch-pc/lab06/lab6-3.asm - Mousepad

Файл Правка Поиск Просмотр Документ Помощь

```
%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.12. Программа lab6-3.asm.

Программа выполняет арифметические вычисления, на вывод идет результирующее выражения и его остаток от деления (рис. 13).

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

Рис. 13. Запуск программы lab6-3.asm.

Заменив переменные в программе для выражения f(x) = (4*6+2)/5 (рис. 14).

~/work/arch-pc/lab06/lab6-3.asm - Mousepad

Файл Правка Поиск Просмотр Документ Помощь

```
%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/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. 14. Изменение программы lab6-3.asm.

Запуск программы lab6-3.asm дает корректный результат (рис. 15).

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

Рис. 15. Запуск измененной программы lab6-3.asm.

Создаю новый файл variant.asm и помещаю текст из листинга и сразу проверяю его наличие в каталоге lab06 (рис. 16-17).

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab06$ touch ~/work/arch-pc/lab06/variant.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/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
```

Рис. 16. Создание файла variant.asm.

~/work/arch-pc/lab06/variant.asm - Mousepad

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 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
call quit
```

Рис. 17. Программа для подсчета варианта.

Запустив программу и указав свой номер студенческого билета, я получила свой вариант для дальнейшей работы (рис. 18).

```
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab06$ nasm -f elf variant.asm
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant variant.o
aryusupova@aryusupova-VirtualBox:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
1032253510
Ваш вариант: 11
```

Рис. 18. Запуск программы для подсчета варианта.

4.3 Ответы на контрольные вопросы.

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

```
moveax,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
```

4.4 Задание для самостоятельной работы

В соответсвии с выбранным вариантом, я реализую программу для подсчета функции f(x)=3(x+10)-20. Для начала создаю файл lab6-4.asm, в котором буду записывать свою программу (рис. 19).

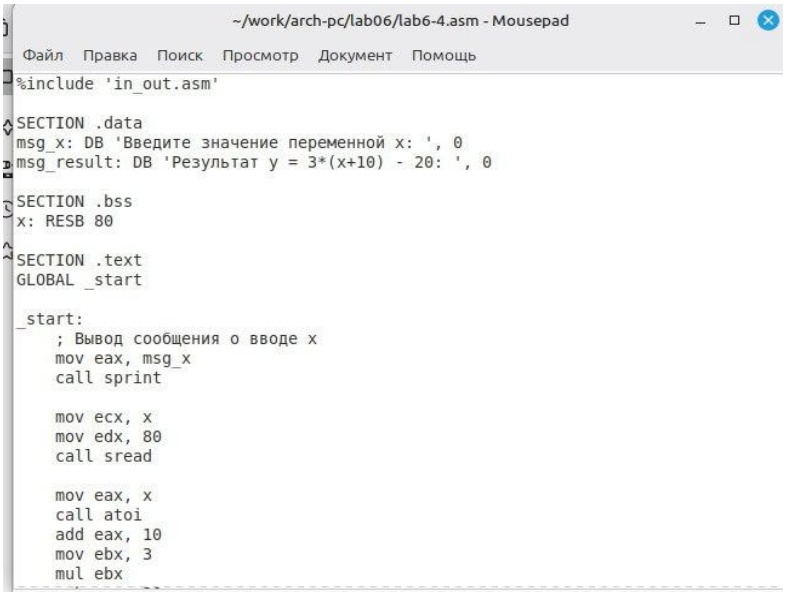

Рис. 19. Написанная мной программа.

Затем проверяю программу на нескольких переменных. Программа выводит верные значения (рис. 20).

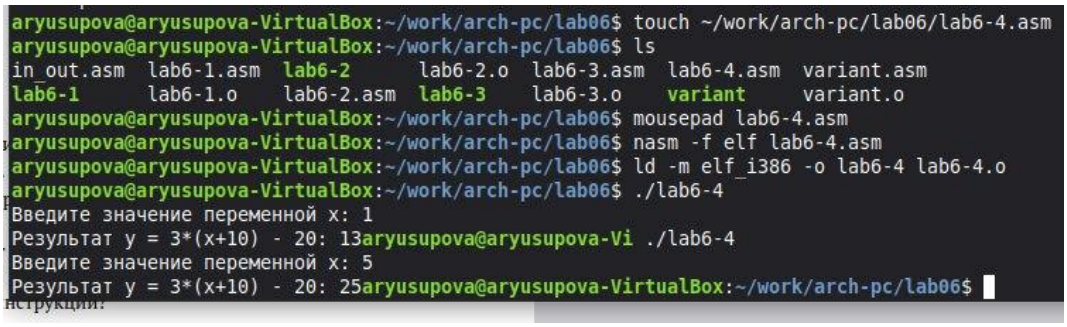

Рис. 20. Запуск и проверка моей программы.

Прилагаю код своей программы:

%include 'in_out.asm'

SECTION .data

msg_x: DB 'Введите значение переменной x: ', 0

msg_result: DB 'Результат y = 3*(x+10) - 20: ', 0

SECTION .bss

x: RESB 80

SECTION .text

```
GLOBAL _start

_start:
    ; Вывод сообщения о вводе x
    mov eax, msg_x
    call sprint
    mov ecx, x
    mov edx, 80
    call sread
    mov eax, x
    call atoi
    add eax, 10
    mov ebx, 3
    mul ebx
    sub eax, 20
    mov edi, eax
    mov eax, msg_result
    call sprint
    mov eax, edi
    call iprint
    mov eax, 1
    xor ebx, ebx
    int 0x80
```

5 Выводы

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

6 Список литературы

1. Ссылка на гитхаб: https://github.com/alyusupova/study_2025-2026_arh-pc.git
2. Руководство по ассемблеру NASM: https://metanit.com/assembler/nasm/
3. NASM Assembly Language Tutorials. — 2021. — URL: https://asmtutor.com/.
4. https://esystem.rudn.ru/course/view.php?id=112