РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 6

Студент: Барханоева Раяна Магометовна

Ст.билет: 1032252468

Группа: НКАбд-01-25

МОСКВА

2025 г

Содержание

1. Цель работы

Освоение арифметических инструкций языка ассемблера NASM.

2. Работа.

Создание каталога (рисунок 1).

```
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc$ mkdir lab06
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc$ cd lab06/
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab06$ touch lab6-1.asm
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab06$
```

Рисунок 1.

Создали каталог и внутри него файл lab6-1.asm.

Программа (рисунок 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 sprintLF
    call quit
```

Рисунок 2.

Добавили программу вывода значения eax.

Тест программы (рисунок 3).

```
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
ld -m elf_i386 -o lab6-1 lab6-1.o
./lab6-1
j
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab06$ ./lab6-1
j
```

Рисунок 3.

В данном случае при выводе значения регистра eax мы ожидаем увидеть число 10. Однако результатом будет символ j. Это происходит потому, что код символа 6 равен 00110110 в двоичном представлении (или 54 в десятичном представлении), а код символа 4 – 00110100(52). Команда add eax,ebx запишет в регистр eax сумму кодов – 01101010 (106), что в свою очередь является кодом символа j

Программа (рисунок 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 sprintLF

    call quit
```

Рисунок 4.

Изменили текст программы и вместо символов записали в регистры числа.

Тест программы (рисунок 5).

```
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab06$ nano lab6-1.asm
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab06$ nasm -f elf lab6-1.asm
ld -m elf_i386 -o lab6-1 lab6-1.o
./lab6-1
```

Рисунок 5.

Создали исполняемый файл и запустили программу. Символ с кодом 10 это перевод строки.

Программа (рисунок 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
```

Рисунок 6.

Тест программы (рисунок 7).

```
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
ld -m elf_i386 -o lab6-2 lab6-2.o
./lab6-2
106
```

Рисунок 7.

В результате работы программы мы получим число 106. В данном случае, как и в первом, команда add складывает коды символов ‘6’ и ‘4’ (54+52=106). Однако, в отличии от программы из листинга 6.1, функция iprintLF позволяет вывести число, а не символ, кодом которогоявляется это число.

Программа (рисунок 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
```

Рисунок 8.

Изменили символы на числа.

Тест программы (рисунок 9).

```
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
ld -m elf_i386 -o lab6-2 lab6-2.o
./lab6-2
10
```

Рисунок 9.

Выводом стало 10.

Программа (рисунок 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
```

Рисунок 10.

Заменили iprintLF на iprint.

Тест программы (рисунок 11).

```
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab06$ nasm -f elf lab6-2.asm
ld -m elf_i386 -o lab6-2 lab6-2.o
./lab6-2
10rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab06$
```

Рисунок 11.

Написав iprint, вместо iprintLF, мы убрали табуляцию.

Программа (рисунок 12.)

```
 GNU nano 7.2                          lab6-3.asm *
%include 'in_out.asm'  ; Подключаем библиотеку с функциями

SECTION .data
    div_msg db 'Результат: ', 0
    rem_msg 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'
    ; ----

    mov eax, div_msg    ; вызов подпрограммы печати
    call sprint         ; сообщения 'Результат: '
    mov eax, edi        ; вызов подпрограммы печати значения
    call iprintLF       ; из 'edi' в виде символов
    mov eax, rem_msg    ; 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
```

Рисунок 12.

Я создала файл lab6-3.asm и прописала туда программу вычисления выражения $f(x) = (5 * 2 + 3)/3$.

Тест программы (рисунок 13).

```
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 la
b6-3.o
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab06$ ./lab6-3
Результат: 4
Остаток от деления: 1
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab06$
```

Рисунок 13.

Создала исполняемый файл и запустила программу.

Программа (рисунок 14).

```
  GNU nano 7.2                     lab6-3.asm *
%include 'in_out.asm'  ; Подключаем библиотеку с функциями

SECTION .data
    div_msg db 'Результат: ', 0
    rem_msg db 'Остаток от деления: ', 0

SECTION .text
GLOBAL _start

_start:
    ; ---- Вычисление выражения
    mov eax, 1          ; EAX=1
    mov ebx, 3          ; EBX=3
    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'
    ; ----

    mov eax, div_msg    ; вызов подпрограммы печати
    call sprint         ; сообщения 'Результат: '
    mov eax, edi        ; вызов подпрограммы печати значения
    call iprintLF       ; из 'edi' в виде символов
    mov eax, rem_msg    ; 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
```

Рисунок 14.

Изменила текст программы для вычисления выражения $f(x) = (4 * 6 + 2)/5$.

Тест программы (рисунок 15.)

```
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab06$ nasm -f elf lab6-3.asm
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab06$ ld -m elf_i386 -o lab6-3 la
b6-3.o
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab06$ ./lab6-3
Результат: 2
Остаток от деления: 0
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab06$
```

Рисунок 15.

Запустили программу с новым выражением.

Программа (рисунок 16).

```
  GNU nano 7.2                     variant.asm *
%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
    call atoi
    xor edx, edx
    mov ebx, 20
    div ebx
    inc edx
    mov eax, rem
    call sprint
    mov eax, edx
    call iprintLF
    call quit
```

Рисунок 16.

Программа которая вычисляет вариант по номеру студенческого билета

Тест программы (рисунок 17).

```
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab06$ ld -m elf_i386 -o variant v
ariant.o
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab06$ ./variant
Введите № студенческого билета:
123456
Ваш вариант: 17
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab06$
```

Рисунок 17.

При вводе номера студенческого билета выдал вариант.

2.1. Ответы на вопросы.

1. Какие строки листинга 6.4 отвечают за вывод на экран сообщения ‘Ваш вариант:’?

mov eax, rem

call sprint

2. Для чего используется следующие инструкции?

movecx, x

movedx, 80

call sread

используются для ввода строки с клавиатуры: `ecx` — адрес буфера для хранения введённых символов, `edx` — максимальное количество символов для чтения, `call sread` — вызов подпрограммы чтения строки.

3. Для чего используется инструкция “call atoi”?

Инструкция `call atoi` используется для преобразования введённых символов ASCII в числовое значение, чтобы над ним можно было выполнять арифметические операции.

4. Какие строки листинга 6.4 отвечают за вычисления варианта?

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

5. В какой регистр записывается остаток от деления при выполнении инструкции “div ebx”?

Остаток от деления при выполнении инструкции `div ebx` записывается в регистр `edx`.

6. Для чего используется инструкция “inc edx”?

Инструкция `inc edx` используется для увеличения остатка на 1, чтобы получить номер варианта в диапазоне от 1 до 20, а не от 0 до 19.

7. Какие строки листинга 6.4 отвечают за вывод на экран результата вычислений?

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

3. Самостоятельная работа

Программа (рисунок 18).

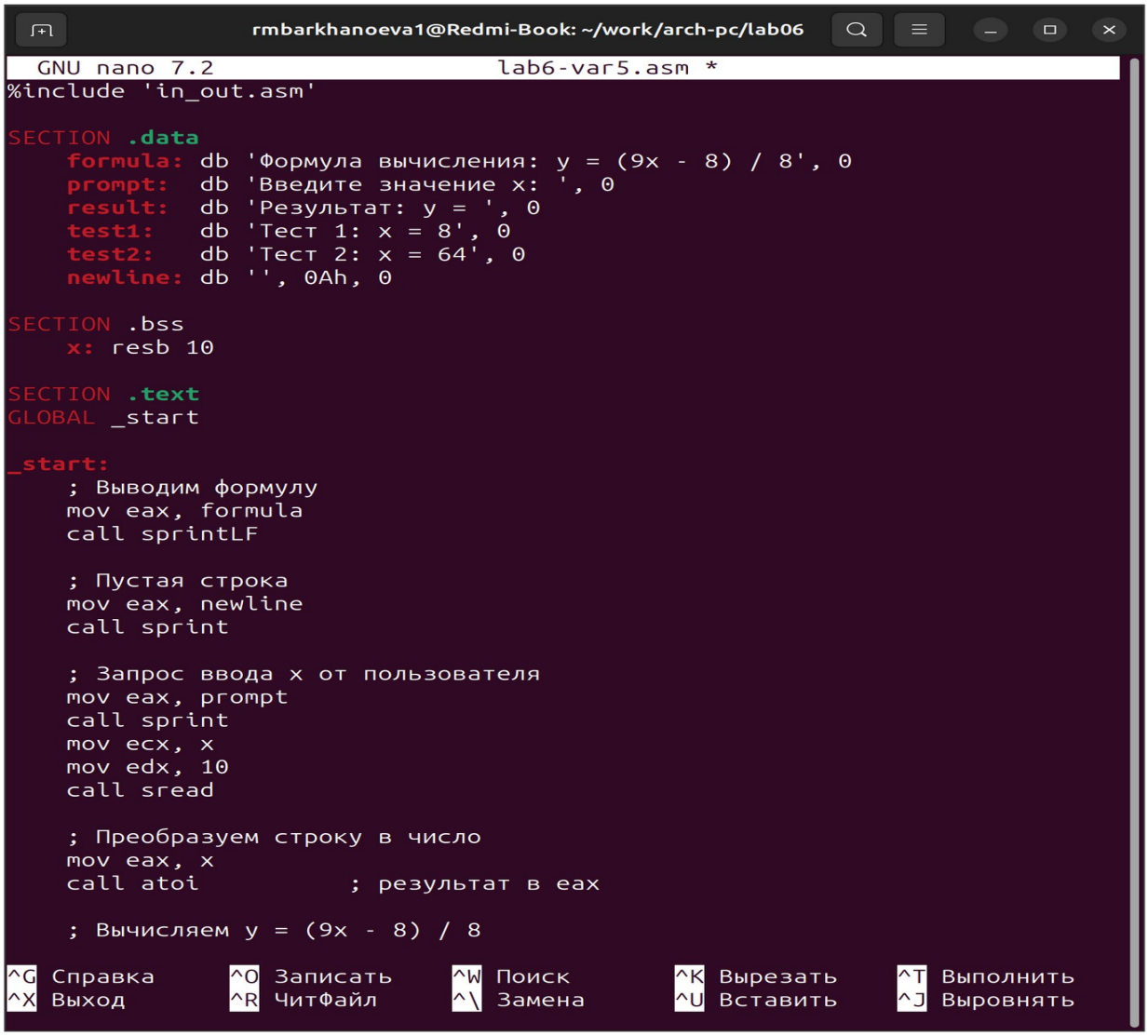

Рисунок 18.

Программа вычисления выражения $y = f(x)$.

Тест программы(рисунок 19).

```
rmbarkhanoeva1@Redmi-Book:~/work/arch-pc/lab06$ nasm -f elf lab6-var5.asm
ld -m elf_i386 -o lab6-var5 lab6-var5.o
./lab6-var5
Формула вычисления: y = (9x - 8) / 8

Введите значение x: 5
Результат: y = 4

Тест 1: x = 8
Результат: y = 8
Тест 2: x = 64
Результат: y = 71
```

Рисунок 19.

Протестировали программу.

Вывод

В ходе лабораторной работы №6 были изучены способы адресации операндов в NASM и освоены основные арифметические инструкции языка ассемблера, включая `add`, `sub`, `inc`, `dec`, `neg`, а также команды умножения `mul`/`imul` и деления `div`/`idiv`. Были рассмотрены особенности работы с символьными и числовыми данными, различия между ASCII-кодами и целыми числами, а также необходимость их преобразования при вводе и выводе информации. На практике отработано использование подпрограмм `iprint`, `iprintLF` и `atoi` для корректного отображения и обработки числовых значений. В результате выполнения лабораторной работы были получены навыки выполнения арифметических вычислений в NASM и понимание принципов работы с данными и регистрами в среде GNU/Linux.