|  |  |
| --- | --- |
| Gerb-BMSTU_01 | ***«*Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ \_\_\_\_\_\_\_ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ\_\_\_\_\_\_\_\_

КАФЕДРА \_\_\_\_\_\_\_\_\_КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**ОТЧЕТ**

**по лабораторной работе № 2**

# Дисциплина: Машинно-зависимые языки и основы компиляции

**Название лабораторной работы:** Программирование целочисленных   
вычислений

Студент гр. ИУ6-42Б **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. Д. Шатский**

(Подпись, дата) (И.О. Фамилия)

Преподаватель  **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ C.C.Данилюк**

(Подпись, дата) (И.О. Фамилия)

Москва, 2023

**Вариант 2.23**

**Цель работы.** Изучение форматов машинных команд, команд   
целочисленной арифметики ассемблера и программирование целочисленных   
вычислений.

**Задание.**

Вычислить целочисленное выражение:

**Программа.**

section .data

EntrMsgb db "Enter b:",10

lenEntrb equ $-EntrMsgb

EntrMsgf db "Enter f:",10

lenEntrf equ $-EntrMsgf

EntrMsgk db "Enter k:",10

lenEntrk equ $-EntrMsgk

EntrMsgn db "Enter n:",10

lenEntrn equ $-EntrMsgn

EntrMsgr db "Enter r:",10

lenEntrr equ $-EntrMsgr

EntrMsgm db "Enter m:",10

lenEntrm equ $-EntrMsgm

ExitMsg db "Answer:",10

lenExit equ $-ExitMsg

section .bss

InBuf resb 10

lenIn equ $-InBuf

OutBuf resb 10

lenOut equ $-OutBuf

X resw 1

B resw 1

F resw 1

K resw 1

N resw 1

R resw 1

M resw 1

section .text

global \_start

\_start:

mov rax, 1

mov rdi, 1

mov rsi, EntrMsgb

mov rdx, lenEntrb

syscall

mov rax, 0

mov rdi, 0

mov rsi, InBuf

mov rdx, lenIn

syscall

mov esi,InBuf

call StrToInt64

cmp EBX, 0

jne StrToInt64.Error

mov [B],ax

mov rax, 1

mov rdi, 1

mov rsi, EntrMsgf

mov rdx, lenEntrf

syscall

mov rax, 0

mov rdi, 0

mov rsi, InBuf

mov rdx, lenIn

syscall

mov esi,InBuf

call StrToInt64

cmp EBX, 0

jne StrToInt64.Error

mov [F],ax

mov rax, 1

mov rdi, 1

mov rsi, EntrMsgk

mov rdx, lenEntrk

syscall

mov rax, 0

mov rdi, 0

mov rsi, InBuf

mov rdx, lenIn

syscall

mov esi,InBuf

call StrToInt64

cmp EBX, 0

jne StrToInt64.Error

mov [K],ax

mov rax, 1

mov rdi, 1

mov rsi, EntrMsgn

mov rdx, lenEntrn

syscall

mov rax, 0

mov rdi, 0

mov rsi, InBuf

mov rdx, lenIn

syscall

mov esi,InBuf

call StrToInt64

cmp EBX, 0

jne StrToInt64.Error

mov [N],ax

mov rax, 1

mov rdi, 1

mov rsi, EntrMsgr

mov rdx, lenEntrr

syscall

mov rax, 0

mov rdi, 0

mov rsi, InBuf

mov rdx, lenIn

syscall

mov esi,InBuf

call StrToInt64

cmp EBX, 0

jne StrToInt64.Error

mov [R],ax

mov rax, 1; системная функция 1 (write)

mov rdi, 1; дескриптор файла stdout=1

mov rsi, EntrMsgm ; адрес выводимой строки

mov rdx, lenEntrm ; длина строки

syscall

mov rax, 0

mov rdi, 0

mov rsi, InBuf

mov rdx, lenIn

syscall

mov esi,InBuf

call StrToInt64

mov [M],ax

mov AX,[B]

imul WORD[B]

mov BX,[F]

sub BX,2

imul BX

idiv WORD[K]

mov BX,AX;

mov AX,[N]

add AX,[R]

imul WORD[M]

sub BX,AX

mov [X],BX

mov rax, 1; системная функция 1 (write)

mov rdi, 1; дескриптор файла stdout=1

mov rsi, ExitMsg ; адрес выводимой строки

mov rdx, lenExit; длина строки

syscall

mov esi, OutBuf ; загрузка адреса буфера вывода

mov ax, [X] ; загрузка числа в регистр

cwde ; развертывание числа из ax в eax

call IntToStr64

mov rdi, 1; дескриптор файла stdout=1

mov rsi, OutBuf ; адрес выводимой строки

mov rdx, rax ; длина строки

mov rax, 1; системная функция 1 (write)

; вызов системной функции

syscall

; read

mov rax, 0; системная функция 0 (read)

mov rdi, 0; дескриптор файла stdin=0

mov rsi, InBuf; адрес вводимой строки

mov rdx, lenIn; длина строки

; вызов системной функции

syscall

; exit

mov rax, 60; системная функция 60 (exit)

xor rdi, rdi; return code 0

syscall

; вызов системной функции

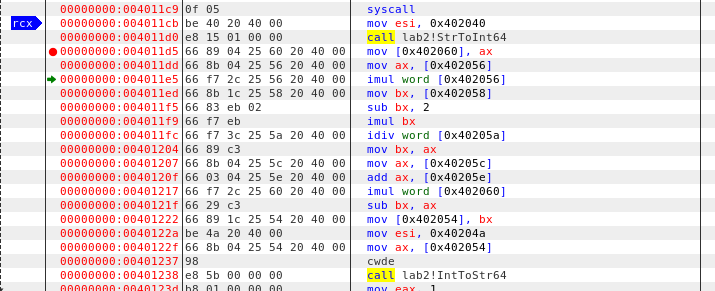
%include "../lib64.asm"

**Тесты.**

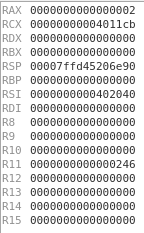
b=2 f=-3 k=2 n=1 r=0 m=2

Выполнение вычислений:

Перемещение в регистр AX параметра B изображено на рисуке 1 и 2.

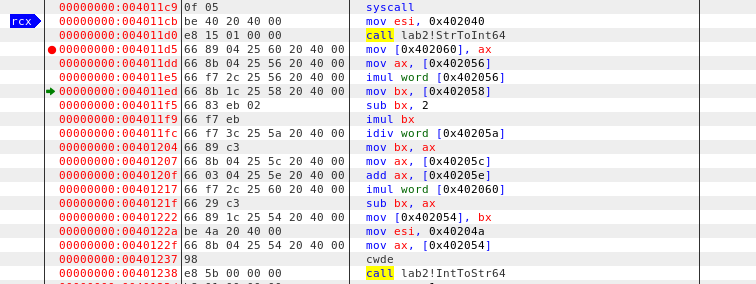


*Рисунок 1 – текущее положение отладчика*

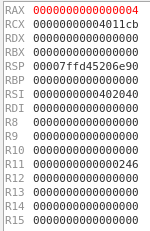


*Рисунок 2 – текущее состояние регистров*

Возведение переменной B в квадрат изображено на рисуке 3 и 4.

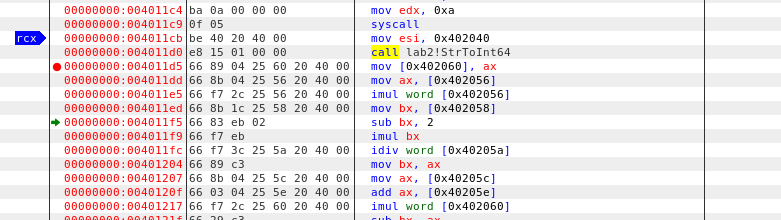


*Рисунок 3 – текущее положение отладчика*

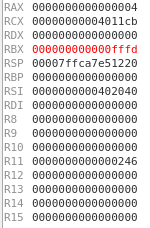


*Рисунок 4 – текущее состояние регистров*

Перемещение в регистр BX параметра F изображено на рисуке 5 и 6.

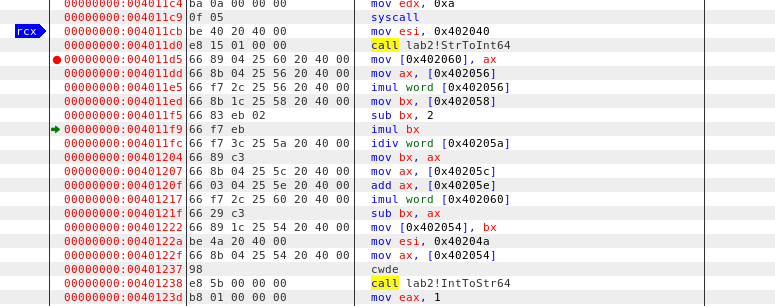


*Рисунок 5 – текущее положение отладчика*

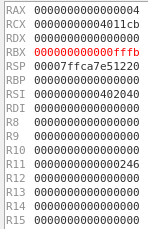


*Рисунок 6 – текущее состояние регистров*

Вычитание из F двойки изображено на рисуке 7 и 8.

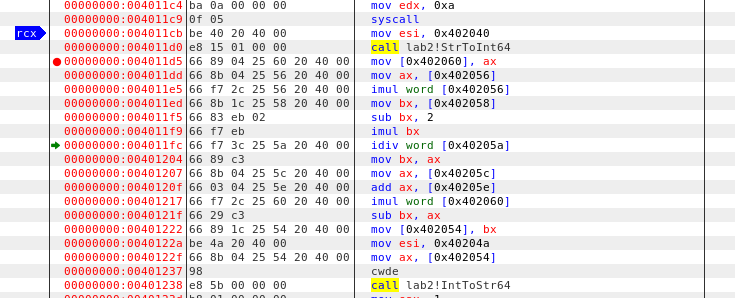


*Рисунок 7 – текущее положение отладчика*

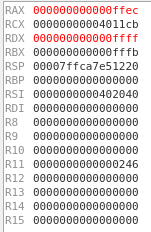


*Рисунок 8 – текущее состояние регистров*

Перемножение B^2 и F-2 изображено на рисуке 9 и 10.

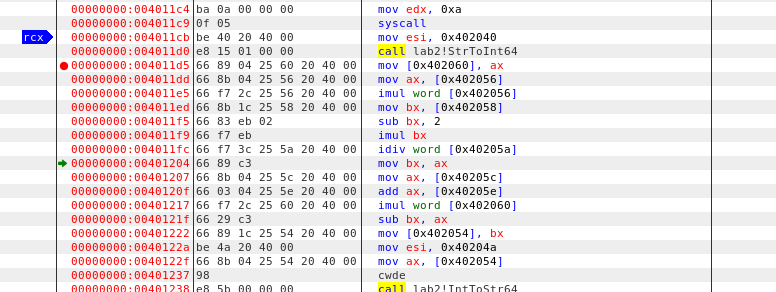


*Рисунок 9 – текущее положение отладчика*

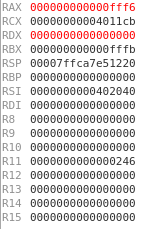


*Рисунок 10 – текущее состояние регистров*

Деление на K изображено на рисуке 11 и 12.

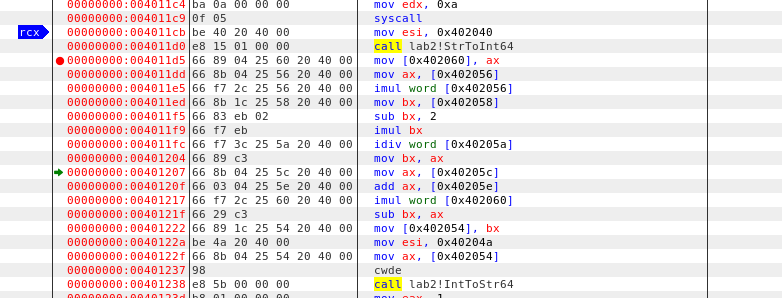


*Рисунок 11 – текущее положение отладчика*

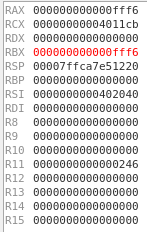


*Рисунок 12 – текущее состояние регистров*

Перемещение результата в BX изображено на рисуке 13 и 14.

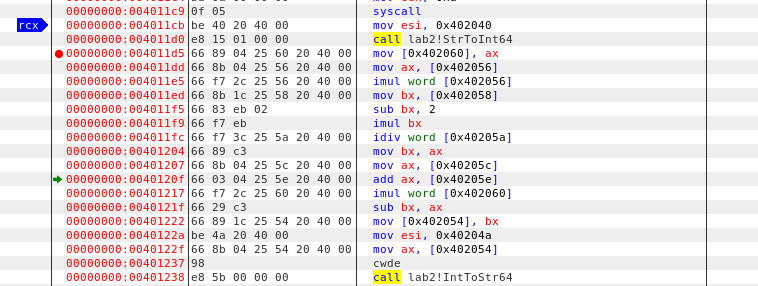


*Рисунок 13 – текущее положение отладчика*

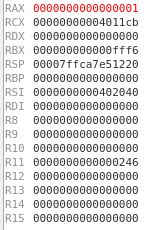


*Рисунок 14 – текущее состояние регистров*

Перемещение в регистр BX параметра N изображено на рисуке 15 и 16.

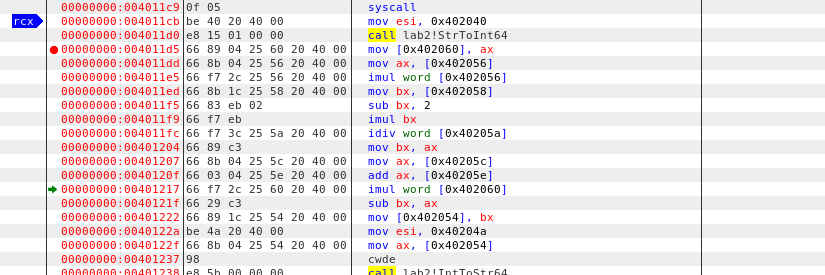


*Рисунок 15 – текущее положение отладчика*

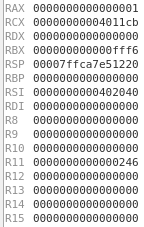


*Рисунок 16 – текущее состояние регистров*

Сложение N и R изображено на рисуке 17 и 18.

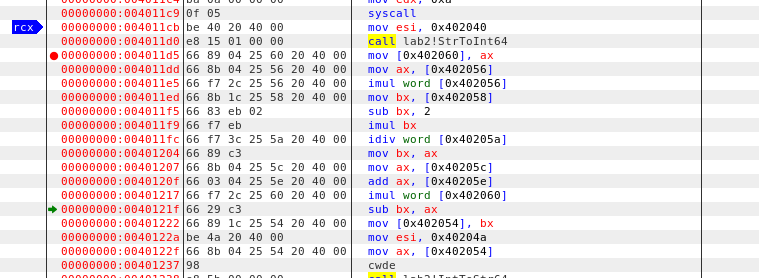


*Рисунок 17 – текущее положение отладчика*

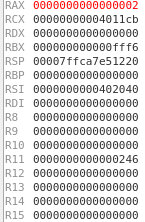


*Рисунок 18 – текущее состояние регистров*

Умножение получившейся суммы на M изображено на рисуке 19 и 20.

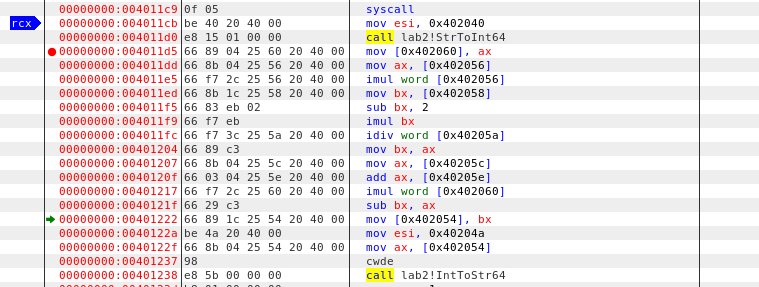


*Рисунок 19 – текущее положение отладчика*

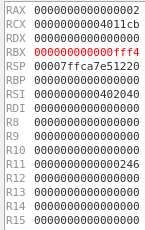


*Рисунок 20 – текущее состояние регистров*

Вычитание из правой части левой части выражения изображено на рисуке 21 и 22.



*Рисунок 21 – текущее положение отладчика*



*Рисунок 22 – текущее состояние регистров*

Расшифровки команд mov.

**66 89 04 25 60 20 40 00 – mov [0x402060],AX**

66 – префикс, определяющий размер операнда (16 бит)

89 = 100010 0 1

100010 – обозначение команды mov

0 – из регистра

1 – размер операнда (16 бит)

04 = 00 000 100

00 – смещение 0 байт

000 – регистр AX

100 – присутсвует байт sib

25 = 00 100 101

00 - масштаб 1

100 – индекс

101 – Disp32 + ss\*index

60 20 40 00 - смещение

**66 8B 04 25 56 20 40 00 - mov AX,[0x402056]**

66 – префикс, определяющий размер операнда (16 бит)

8B = 100010 1 1

100010 – обозначение команды mov

1 – в регистр

1 – размер операнда (16 бит)

04 = 00 000 100

00 – смещение 0 байт

000 – регистр AX

100 – присутсвует байт sib

25 = 00 100 101

00 - масштаб 1

100 – индекс

101 – Disp32 + ss\*index

56 20 40 00 - смещение

**66 89 C3 – mov BX,AX**

66 – префикс, определяющий размер операнда (16 бит)

89 = 100010 0 1

100010 – обозначение команды mov

0 – из регистра

1 – размер операнда (16 бит)

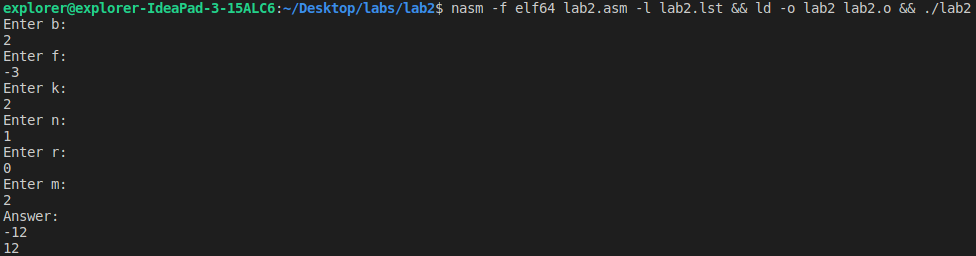
C3 = 11 000 011

11 – из регистра в регистр

000 – регистр AX

011 – регистр BX

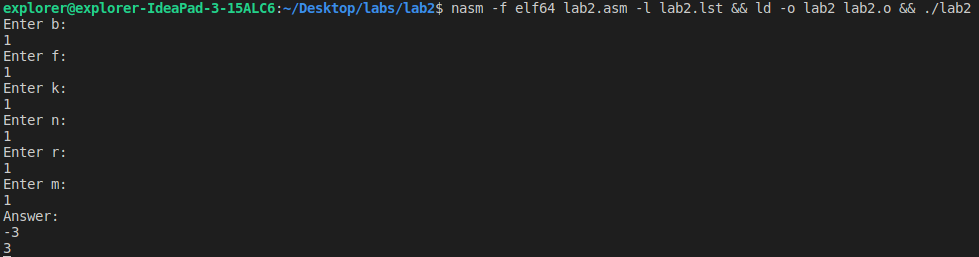
Результат выполнения вычислений программы можно посмотреть на рисунке 23.



*Рисунок 23 – результат тестирования*

b=1 f=1 k=1 n=1 r=1 m=1

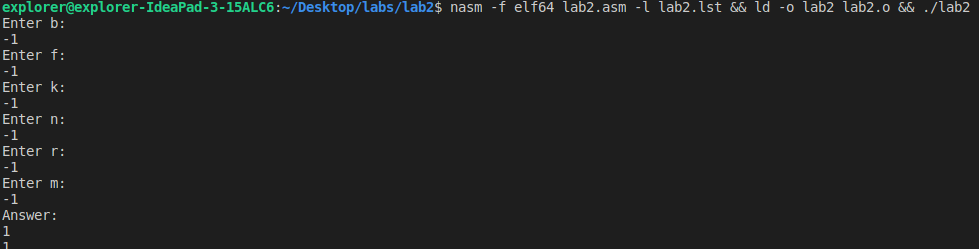
Результат выполнения вычислений программы можно посмотреть на рисунке 24.



*Рисунок 24 – результат тестирования*

b=-1 f=-1 k=-1 n=-1 r=-1 m=-1

Результат выполнения вычислений программы можно посмотреть на рисунке 25.

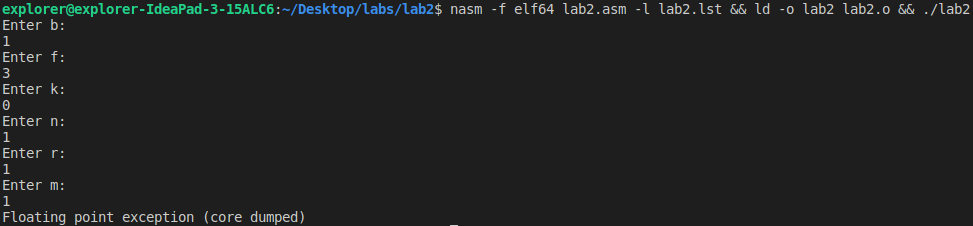


*Рисунок 25 – результат тестирования*

b=1 f=3 k=0 n=1 r=1 m=1

Проверка ошибки деления на ноль

Результат выполнения вычислений программы можно посмотреть на рисунке 26.

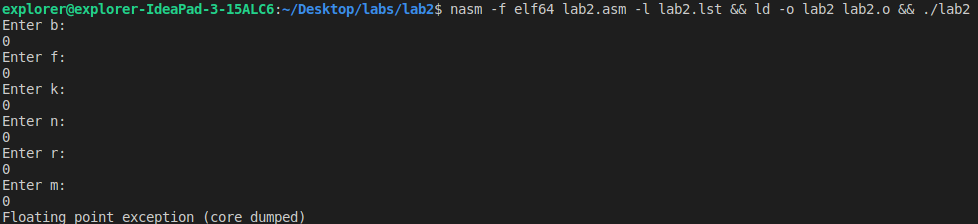


*Рисунок 26 – результат тестирования*

b=0 f=0 k=0 n=0 r=0 m=0

Проверка ошибки деления на ноль

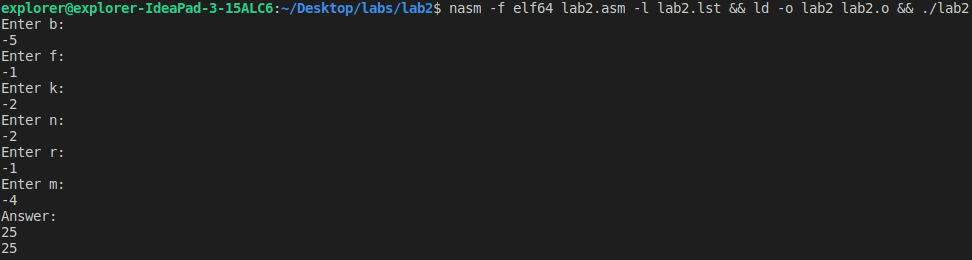
Результат выполнения вычислений программы можно посмотреть на рисунке 27.



*Рисунок 27 – результат тестирования*

b=-5 f=-1 k=-2 n=-2 r=-1 m=-4

Результат выполнения вычислений программы можно посмотреть на рисунке 28.



*Рисунок 28 – результат тестирования*

**Контрольные вопросы.**

**1. Что такое машинная команда? Какие форматы имеют машинные команды процессора IA32? Чем различаются эти форматы?**

Машинная команда – это код, дающий указания процессору какие действия нужно выполнять. Размер машинной команды различается для каждой отдельной команды. Также команды могут иметь различные префиксы, например 66h и 67h и другие. **2. Назовите мнемоники основных команд целочисленной арифметики.   
Какие форматы для них можно использовать?**

ADD – Сложение двух операндов и помещение результата по адресу первого

Форматы: REG, MEM;MEM, REG;REG, REG;MEM, IMM;REG, IMM

SUB – Вычитание двух операндов и помещение результата по адресу первого

Форматы: REG, MEM;MEM, REG;REG, REG;MEM, IMM;REG, IMM

DIV – Разделить то что находится в регистре DX:AX (или AX, EDX:EAX в зависимости от разрядности операции) на операнд. Результат помещается в регистр AX (AL,EAX), остаток в регистр DX(AH, EDX).

Форматы: REG;MEM

MUL – Умножить то что находится в регистре AX (или AL, EAX в зависимости от разрядности операции) на операнд. Результат помещается в регистр DX:AX (AX,EDX:EAX).

Форматы: REG;MEM

INC – Увеличить операнд на 1

Форматы: REG;MEM

DEC – Уменьшить операнд на 1

Форматы: REG;MEM

**3. Сформулируйте основные правила построения линейной программы   
вычисления заданного выражения.**

Для вычисления заданого выражения на ассемблеренеобходимо выбрать регистр-аккумулятор и записать в него результат вычисления одной из частей выражения. Затем необходимо последовательно выполнять операции и записывать получившиеся значения в аккумулятор. **4. Почему ввод-вывод на языке ассемблера не программируют с исполь-  
зованием соответствующих машинных команд? Какая библиотека используется для организации ввода вывода в данной лабораторной?**

На языке ассемблера невозможно запрограмировать ввод-вывод, так как для начала нужно преобразовать все данные в символьный вид. Для этого используется библиотека lib64.asm. **5. Расскажите, какие операции используют при организации ввода-  
вывода.**

Для вывода:

mov rax, 1; системная функция write

mov rdi, 1; дескриптор файла

mov rsi, OutBuf ; адрес выводимой строки

mov rdx, lenOut ; длина выводимой строки

syscall

Для ввода:

mov rax, 0; системная функция read

mov rdi, 0; дескриптор файла

mov rsi, InBuf; адрес вводимой строки

mov rdx, lenIn; длина вводимой строки

syscall

**Вывод.** Былиизучены форматы машинных команд, команд   
целочисленной арифметики ассемблера и программирование целочисленных   
вычислений. Программа работает корректно на заданныъ тестовых данных.