**Лабораторная работа по информатике №1**

**Цель работы**: понять базовые принципы выполнения арифметических операций и работы с регистрами в архитектуре x86 на языке ассемблера NASM.

**Задание 1**

Для заданной формулы ***a+b/c***в программу поступают 2 набора входных значений: A = {4, 23, 21}, B = {6, 13, -5}. Набор A не требует перевода в дополнительный код, так как все его значения беззнаковые. Дополнительный код для набора B в HEX будет выглядеть следующим образом: {0x06, 0x0D, 0xFFFB}. Он строится переводом числа в двоичную систему счисления, инвертированием всех бит и прибавлением 1. Пример для числа -5:

5 => 00000101b => 11111010b => 11111011b => 0xFFFBh => -5

**Листинг**

section .text

global CMAIN

CMAIN:

MOV EBP, ESP

1. MOV AX, 0xFF00
2. MOV AL, [b]
3. MOV BL, [a]
4. MOV CL, [c]
5. CBW
6. IDIV CL
7. MOV BL, [a]
8. ADD BL, AL
9. ret

section .data

a db 0x06 ; two’s complement of 6

b db 0x0D ; two’s complement of 13

c db 0xFFFB ; two’s complement of -5

**Анализ**

Для корректной работы со знаковыми числами старший байт регистра AX заполняется единицами, а в младший помещается значение переменной *b.* В регистр BX записывается значение *a.* Операция IDIV делит регистр AX на CL и записывает частное в AL, а остаток в AH.

Трассировочная таблица для набора B выглядит следующем образом:

|  |  |  |
| --- | --- | --- |
| № | AX | BX |
| 1 | 0xFF00 | 0x0000 |
| 2 | 0xFF13 | 0x0000 |
| 3 | 0xFF13 | 0x0006 |
| 4 | 0xFF13 | 0x0006 |
| 5 | 0x04FD | 0x0003 |

После выполнения программы в регистре AL остается частное выражения, равное *-3*, а в регистре AH – остаток, равный FFFB = *3*. Результаты являются верными для выражения 6+13/(-5), следовательно программа работает правильно.

**Задание 2**

Для заданной формулы ***a+b/c*** в программу поступают 2 набора входных значений: A={4, 23, 21}, B={6, 13, -5}. Для получения результата в вещественном виде все числа необходимо перевести в формат IEEE754. Переведем числа из набора B: {0x40С00000, 0x41500000, 0xC0A00000}. Алгоритм перевода в IEEE754 на примере числа *-5* следующий:

1. Число переводится в BIN и приводится к виду нормализованной мантиссы и экспоненты (101 => мантисса — 1.50, экспонента — 2)
2. К значению экспоненты прибавляется 127 (экспонента — 129 = 10000011b)
3. Знаковый бит равен 1, т.к. число отрицательное.
4. 32 бит занимает знаковый, 31-24 биты занимает экспонента, оставшиеся биты — дробная часть мантиссы (11000000 10100000 00000000 00000000b = 0xC0A00000)

Для записи выражения в стек используем обратную польскую нотацию: *abc+/*.

**Листинг**

section .text

global CMAIN

CMAIN:

MOV EBP, ESP

1. FLD DWORD [b]
2. FLD DWORD [a]
3. FLD DWORD [c]
4. FDIV
5. FADD
6. FST DWORD [result]
7. ret

section .data

a DD 0x40С00000 ;IEEE754 of 6

b DD 0x41500000 ;IEEE754 of 13

c DD 0xC0A00000 ;IEEE754 of -5

result DD 0x00000000 ;empty variable

**Анализ**

Командой FLD запишем переменные. FADD и FDIV произведут сложение и деление соответственно. Полученный результат запишем в пустую переменную *result* командой FST.

**Задание 3**

**Номер 2**

Необходимо составить программу, определяющую по сторонам треугольника, является ли он равнобедренным (с учетом того, что треугольник существует). Алгоритм определения для треугольника со сторонами *a,b,c* будет выглядеть следующим образом:

**Листинг**

section .text

global CMAIN ;the program checks if a triangle is isosceles

CMAIN:

1. MOV EBP, ESP
2. MOV AL, [a]
3. MOV BL, [b]
4. CMP AL, BL
5. JE YES
6. CMP AL, [c]
7. JE YES
8. CMP BL, [c]
9. JE YES
10. MOV AX, 0
11. ret
12. YES:
13. MOV AX, 1

ret

section .data

a db 3

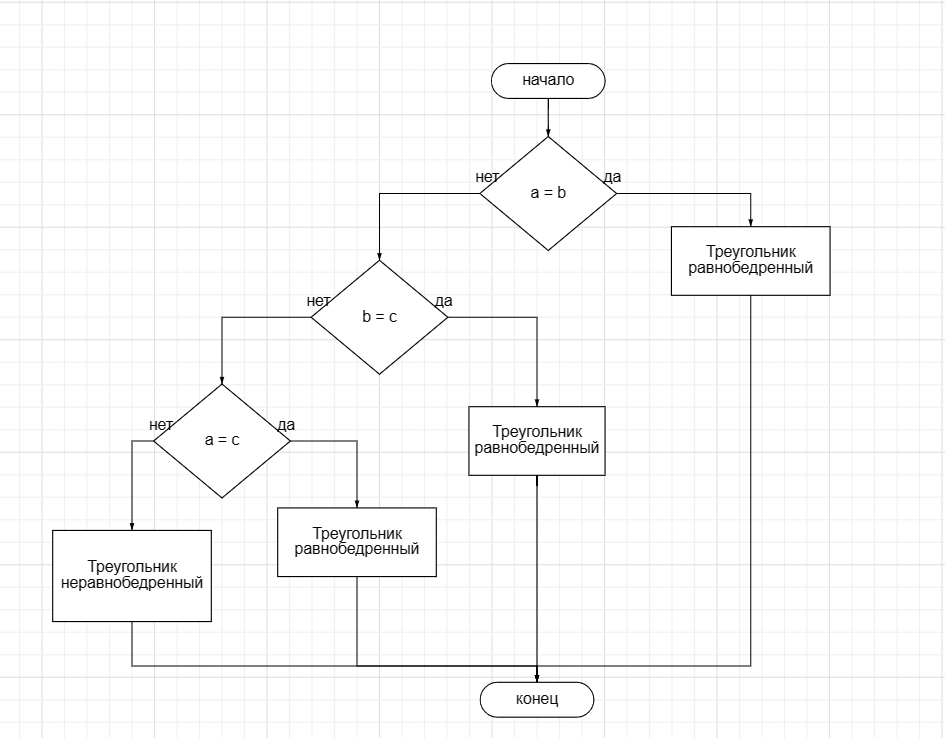
b db 5

c db 3

**Анализ**

В регистры AL и BL помещаются стороны *a* и *b.* Они попеременно сравниваются друг с другом и со стороной *c*, записанной в переменной с. Результатом программы является число, записанное в AX – 1, если треугольник равнобедренный, 0, если нет.

Для набора {3, 5, 3} после выполнения программы в регистре AX окажется 1, что соответствует справедливости равнобедренности треугольника с заданными сторонами, а для набора {6, 8, 10} в AX окажется 0.



**Вывод**

Я выполнил все задания и понял базовые принципы работы на языке ассемблера, проведя арифметические операции разных типов и составив работающий алгоритм.