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

В первом задании программа получает на вход тестовые 8-битные значения **A = {10, 26, 27}**, **B = {-4, -22, 6}**. Затем вычисляет их по формуле **a/(b-c)**. Для деления со знаком все числа переводятся в дополнительный код:

{A}

10 = 0x0A

26 = 0x1A

27 = 0x1B

{B}

-4 = 0xFC

-22 = 0xEA

6 = 0x06

Доп. код строится переводом числа в двоичную систему счисления, инвертированием всех бит и прибавлением 1. Для записи в регитры используется hex-запись.

# Листинг

Код программы приведен в файлах **lab1\_1.asm** и **lab1\_2**.**asm**.

# Анализ

Для корректной работы со знаковыми числами старший байт регистра AX (AH) заполняется **нулями**, так как в AX записываются положительные числа, а в младший (AL) помещается значение перменной **a***.* В регистр BX записывается значение**b***,* а затем разность с переменной**c***.* Операция **IDIV** делит регистр AX на BL и записывает частное в AL, а остаток в AH.

# Пример

Для набора A после выполнения программы в регистре AL остается частное выражения, равное **-10**, а в регистре AH – остаток, равный **0**. Результаты являются верными для выражения 10 / (26 - 27), следовательно программа выполняется правильно.

# Задание 2

Для заданной формулы a / (b-c) в программу поступают те же наборы входных значений: A = {10, 26, 27}, B={-4, -22, 6}. Для получения результата в вещественном виде все числа необходимо перевести в формат **IEEE754**. Перевод выполняется следующим образом (На примере A[0] = 10):

1. Число переводится в **bin** и приводится к виду нормализованнной мантиссы и экспоненты *(Для 1010b: мантисса — 1.01b, экспонента — 3)*
2. К значению экспоненты **прибавляется 127** *(экспонента — 130 = 10000010b)*
3. **Знаковый бит** равен 0, т.к. число положительное.
4. 32-ой бит занимает знаковый, 31-24 биты занимает экспонента, оставшиеся биты — дробная часть мантиссы *(01000001001000000000000000000000b = 0x41200000)*

Для записи выражения в стек используется **обратную польскую нотацию**: a b c - / .

# Листинг

Код программы приведен в файлах **lab2\_1.asm** и **lab2\_2.asm**.

# Анализ

Командой **FLD** записываются переменные. **FSUB** и **FDIV** производят вычитание и деление соответственно. Полученный результат запишем командой **FST** в пустую переменную **result** размера **double word** *(0x00000000)*.

# Пример

Рассмотрим трассировочную таблицу стека для набора A:

|  |  |  |
| --- | --- | --- |
| № | Оставшаяся цепочка | Стек |
| 0 | 0x41200000, 0x41D00000, 0x41D80000, -, / |  |
| 1 | 0x41D00000, 0x41D80000, -, / | 0x41200000 |
| 2 | 0x41D80000, -, / | 0x41200000, 0x41D00000, |
| 3 | -, / | 0x41200000, 0x41D00000, 0x41D80000 |
| 4 | / | 0x41200000, 0xBF800000 |
| 5 |  | 0xC1200000 |

После выполнения программы в переменную **result**запишется значение 0xC1200000, что соответсвует IEEE754-записи числа *-*10. Результаты являются верными для выражения 10 / (26 - 27), следовательно программа выолняется правильно.

# **Задание 3 (№2)**

Программа получает на вход три значения a, b, c, являющиеся **сторонами треугольника**. Результатом ее выполнения будет установление **равнобедренности** треугольника (с учетом того, что треугольник **существует**). Алгоритм выглядит следующим образом *(рис. isosceles triangle)*:

# Листинг

Код программы приведен в файле **lab3**.**asm**

# Анализ

В регистры AL и BL загружаются значения сторон **a** и **b***.* Они сравниваются командой **CMP** друг с другом и со стороной **c** каждая попеременно. Результатом программы является выводимое макросом **PRINT\_DEC** число:

**1** — если треугольник равнобедренный

**0** — если труегольник неравнобедренный

# Пример

Для набора **{1,2,1}** результатом выполения программы будет выведенная 1, а для набора **{3,4,5}** – 0. Следовательно программа выполняется правильно.

# Вывод

После выполнения всех заданий я понял базовые принципы работы на языке ассемблера, а также разобрался в проведении арифметических операций и реализации базовых алгоритмов.