**Лабораторная работа по теме 3 – Типы данных и машинная арифметика**

Группа РИС-22-1, ФИО-Мусихин Д. М.

**Задачи лабораторной работы – часть 1**

1. Сколько раз выполнится цикл в программе, фрагмент кода которой на языке Pascal приведён ниже, если переменная *S* имеет *вещественный тип однократной точности* *l*). Какое значение получит переменная *S*?

S := 1/3;

while S <> 1 do {Выполнять, пока S не равно 1}

S := S + 1/3;

Поясните ответ – выполните трассировку программы (пошаговое выполнение) – заполните таблицу, структура которой приведена ниже, чтобы обосновать свой ответ (покажите, как меняется значение переменной):

|  |  |  |  |
| --- | --- | --- | --- |
| Оператор/ операция | **Десятичное значение переменной *S*** | Код (**внутреннее представление значения переменной *S* в памяти компьютера**) в выбранном формате | Комментарий |
| S := 1/3; | 0,333333333 | 0 01111110 01010101010101010101010 | Вычислите значение *S* |
| S <> 1 | 0,333333333 | 0 01111110 01010101010101010101010 | Условие выполняется |
| S := S + 1/3 | 0,666666666 | 0 01111111 01010101010101010101010 | Тело цикла выполняется |
| S <> 1 | 0,666666666 | 0 01111111 01010101010101010101010 | Условие выполняется |
| S := S + 1/3 | 1 | 0 01111111 11111111111111111111111 |  |
| S <> 1 | 1 | 0 01111111 11111111111111111111111 | Условие не выполняется.  Производится выход из цикла.  Из этого пункта можно увидеть, что когда число доходит до последовательности 11111…, то процессор автоматически представляет это число, как просто 1. |

**Примечание:** В этом задании можно было научиться понимать, как процессор работает с вещественными числами. Для себя я понял, что даже компьютер может ошибаться в счете с вещественными числами, так как у него есть ограничения для той же мантиссы и порядка, например. Из данного случая я узнал, каким образом 1/3 + 1/3 + 1/3 = 1 для компьютера, и что 1 не всегда является таковой.

Изображение выглядит как текст, снимок экрана, меню

Автоматически созданное описание

Рисунок 1 - Дисассемблированный код для float

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 2 - Дисассемблированный код для double

Аналог кода на питоне:

S = 1/3

while S != 1:

S += 1/3

1. Сколько раз выполнится цикл в программе, фрагмент кода которой на языке C++ приведён ниже, если переменная *A* имеет целочисленный тип – *целое без знака в формате байта* (контроль выхода за допустимый диапазон значений отключён). Какое значение пример переменная *A* после завершения цикла?

static unsigned char A;

A = 255;

{ A++;

} while (A != 0);

Поясните ответ – выполните трассировку программы (пошаговое выполнение) – заполните таблицу, структура которой показана ниже, чтобы обосновать свой ответ (покажите, как меняется значение переменной):

|  |  |  |  |
| --- | --- | --- | --- |
| Оператор/ операция | **Десятичное значение переменной *A*** | Код (**внутреннее представление значения переменной *A* в памяти компьютера**) в выбранном формате | Комментарий |
| A = 255; | 255 | 1111 1111 |  |
| A++ | 0 | 0000 0000 |  |
| A != 0 | 0 | 0000 0000 | Условие не выполняется |
| A++ | 0 | 0000 0000 | Цикл не повториться |

**Примечание:** Из этого задания можно было узнать, что после переполнения данного числу (>=0) количества байт, оно становится нулем, и как это происходит.

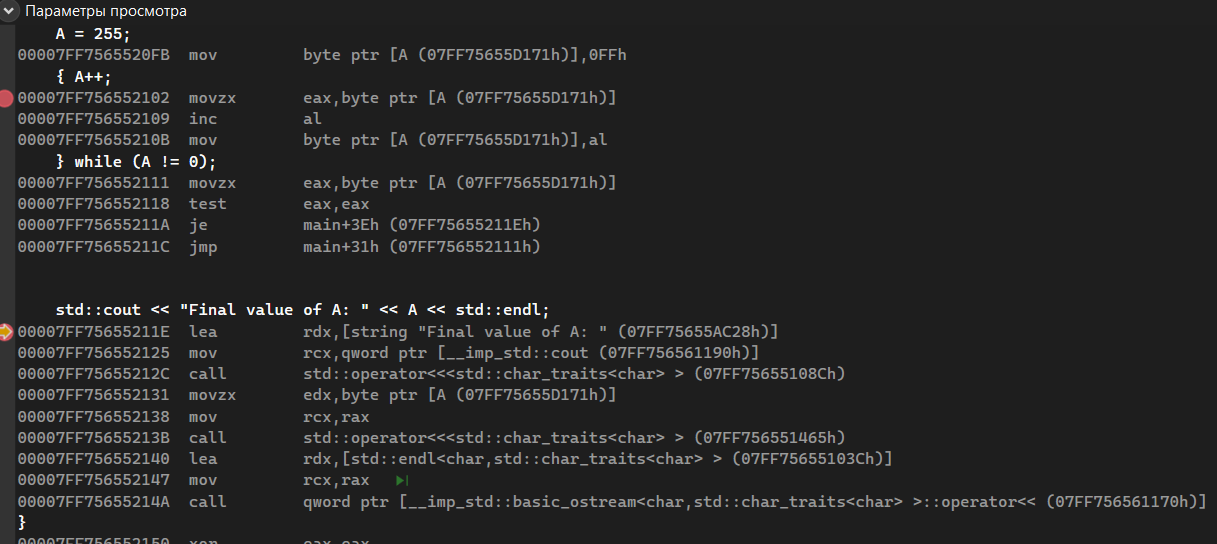


Рисунок 3 - Дисассемблированный код

Аналог кода на С#:

static unsigned char A;

A = 255;

do {

A++;

} while (A != 0);

1. Сколько раз выполнится цикл в программе, фрагмент кода которой на языке C++ приведён ниже, если переменная *A* имеет целочисленный тип (контроль выхода за допустимый диапазон значений отключён):

static signed \_\_int8 A;

A = -127;

while (A < 0)

{ A = A - 1;

}

Поясните ответ – выполните трассировку программы (пошаговое выполнение) – заполните таблицу, чтобы обосновать свой ответ (покажите, как меняется значение переменной):

|  |  |  |  |
| --- | --- | --- | --- |
| Оператор/ операция | **Десятичное значение переменной *A*** | Код (**внутреннее представление значения переменной *A* в памяти компьютера**) в выбранном формате | Комментарий |
| A = -127; | -127 | 1 000 0001 |  |
| A < 0 | -127 | 1 000 0001 | Условие выполняется |
| A = A - 1; | -128 | 1 000 0000 | Тело цикла выполнится |
| A < 0 | -128 | 1 000 0000 | Условие выполняется |
| А = А – 1; | 127 | 0 111 1111 | Из этого перехода можно увидеть, почему после вычитания из минимально числа получается максимально возможное в представлении в виде байта в компьютере |
| A < 0 | 127 | 0 111 1111 | Условие не выполняется |

**Примечание:** Из этого задания можно было узнать, как происходит работа с отрицательными числами, и почему после вычитания из минимально возможного числа для байта (да и для любого числа байт), число превращается максимально возможное.

Изображение выглядит как текст, снимок экрана

Автоматически созданное описание

Рисунок 4 - Дисассемблированный код

Аналог кода на C#:

static signed byte A;

A = -127;

while (A < 0)

A = A - 1;

**Итоги:** Обобщая все вышеописанное, хочу сказать, что для себя я понял, каким образом разные виды чисел хранятся в компьютере, что компьютер не всегда бывает точным, что у него есть ограничения по вычислениям и что эти знания, действительно, необходимы для полноценной работы с вычислительными машинами.

1. В 2СС:

А) 4710 = 1011112  Б) 1210 = 11002 В) 0.87510 = 0.1112 Г) 21.7410 = 10101.1011110...2