**в)** Измените тип всех использованных вами в функции strToInt локальных переменных на int64\_t и перекомпилируйте программу. Опишите, как изменился дизассемблерный листинг.

Странно, что изменился контекст вызова функции-ведь мы меняли размер переменных только внутри неё. Видимо, нам пришлось выделить для функции больше регистров, а раз так вышло, то надо, прежде чем эти регистры отдавать под использование, «залить» их содержимое в стек.

Конечно, изменился стековый кадр функции. Увеличился он на 8 байт, что странно, ибо 4 переменных изменились с int32 на int64 и предполагалось, что и стековый кадр увеличится на 4\*4=16 байт.

Для хранения переменных number и power теперь выделяется по 2 регистра, в отличие от одного в случае с int32.

28: for(int64\_t i=lenstr\_crutch-1;i>=0;i--)//reading the number backwards

В первом случае в один регистр загружалась вычисленная длина, а потом она переносилась с декрементированием в другой регистр. Во втором же длина заносилась в регистр, там же происходило декрементирование, а потом значение копировалось в другой регистр.

Уже более интересное замечание: инструкция CMP была заменена на SUBS и SBCS. Первая из них вычитает из правой части (последние 4 байта по big-endian) счётчика правую часть «эталонного» числа с поднятием флагов. А вторая-соответственно левую. С поднятием и учётом флагов. Условие перехода BGE по сути не изменилось.

Во внутреннем цикле логика та же. Условие перехода одинаковое (BLT), так как флаги расставляются по одинаковому принципу, то есть через вычитание.

Сравнение символов не изменилось, так как символы весят всё тот же байт.

Что удивительно: при обратном перемещении stack pointer на величину стекового кадра в программе с int64 появляются ранее ожидаемые 34байта. Но при этом POP возвращает те же регистры, но без одного. То есть складывается ощущение, что два байта из стекового кадра были перенесены под видом видом контекста вызова. Тем не менее баланс взятого и отданного присутствует)