Задание

Составить на ассемблере БЭВМ программу, которая по поступлению сигнала готовности с ВУ по выбору затирает содержимое памяти. После завершения работы программы как можно больше ячеек памяти должны быть равны 0000, независимо от их предыдущего значения.

Решение.

Единственный способ стереть данные из ячейки – записать туда ноль. В общем случае это можно сделать только командой ST, если в аккумуляторе 0. Но аккумулятор можно очистить всего один раз и дальше это можно не делать, ведь загружать туда мы ничего не собираемся.

Понятно, что основная проблема у нас будет стереть саму программу, а всё остальное должно стараться в цикле. Последней командой в любом случае будет HLT, чтобы наш скрипт не гулял по операциям NOP до бесконечности. Но, чтобы после каждого цикла мы переходили на следующий нам нужна команда ветвления. Причём, когда мы сотрём уже всё по максимуму, будет нужно, чтобы эта команда не сработала. Т.е. мы либо удалим её к этому моменту (1), либо подберём числа (2) так, чтобы команда пропустила переход по циклу.

1. Попробуем способ, когда команду ветвления мы удаляем, тогда, когда последний раз срабатывает команда ST, она стирает команду ветвления, чтобы дальше перейти на HLT.

Чтобы стирать разные ячейки у нас так же должен быть счётчик, указывающий на ячейку, которую мы хотим обнулить. Саму ячейку с указателем нам тоже придётся обнулять, но вот нулевое значение можно было бы использовать и поставить последнюю удаляемую команду на адрес 0x0, но тогда после удаления по адресу указателя будет не ноль, а минус единица. Плюс такое размещение программы помешало бы другим программам использовать стек и прерывания, ведь наша программа была бы размещена в области начала стека и на первых двух векторах прерывания. Тогда мы получается, что минимум команд, которые в любом случае останутся – это 3 команды: HLT и ST, POINT. (Здесь и дальше оставшиеся не стёртыми команды выделены **полужирным.**)

```
                ORG     0x7EB
7FB     WAIT:   IN      3
7FC             AND     #0x40
7FD             BEQ     WAIT
7FE             CLA
7FF     REPITE: ST      -(POINT)
000             BR      REPITE
001             HLT
002     POINT:  WORD    0x7FF
```

Теперь можно попробовать избавиться от проблемы с тем, что такая программа завязана на конкретные места и не перемещаемы. Выделим отдельный BR команды HLT и ждём пока он обнулится, а значит пропустится и мы попадём на HLT вместо нового круга. Т.е. воспользуемся тем же принципом – стиранием BR.

Мне кажется, что среди приведённых примеров – этот лучший, именно, потому что реально физический действий, чтобы очистить ячейки в этом и предыдущем случае уйдёт примерно

одинаково, но здесь программист не будет стеснён отсутствием возможности использовать стек и прерывания в других программах.

```
          ORG       0x50
WAIT:     IN        3
          AND       #0x40
          BEQ       WAIT
          CLA
REPITE:   BR        CLEAR
          HLT
CLEAR:    ST        (POINT)+
          BR        REPITE
POINT:    WORD      0x59
```

2. Но вернёмся к варианту, когда мы подбираем числи и условия, чтобы остановить работу программы.

Мы могли бы подобрать количество циклов в команде LOOP так, чтобы, во-первых, указатель совпадал с номером удаляемой ячейки, во-вторых, ячейкой с номером 0 была сама ячейка с ST. Тогда мы после завершения программы указатель будет пустой, и его обнулять будет не надо. Но из-за того, что команды в БЭВМ исполняются от меньшего номера к большему мы не можем удалить первую команду в цикле удаляя по указателю. Поэтому так LOOP использовать не получится.

Можно было бы попробовать вместо обычного BR писать какое-то условие, но тогда переход займёт больше команд, которые мы скорее всего не сможем удалить, и они останутся после завершения работы программы. Ведь команда CMP, необходимая для установки признаков результата перед другими командами ветвления, меняет значение аккумулятора, т.е. нам придётся загружать что-то в аккумулятор, чтобы сравнить и чистить его после операции.

Вывод: из предложенных размышлений видно, что сделать программу, случайном месте памяти и удаляющей все кроме ≤ 3 команд не выходит. В то время как, если разрешить программе располагаться в любом месте памяти (т.е. дать программисту выбрать наиболее удобное место) можно обойтись тремя оставшимися командами.