ARM Assembly Language programs

Execute the following ARM assembly programs in keil uVision. Simulate in single step mode and show the output.

1. Divide a 8 bit variable into two 4 bit nibbles and store one nibble in each byte of a 16 - bit variable. Store the disassembled byte in memory location (pointed by result)
2. Compare 2 values stored in memory location and store the higher value in a memory location (pointed by Result).
3. Write a program to add two 64 bit numbers and store the result in a memory location.
4. Write an assembly program to evaluate the following expressions without using MUL instruction.
    a. `f(x) = (291*x + 5) / 16`
    b. `f(x) = 595*x + 19`

    Hint : When multiplying by a constant value, it is possible to replace the general multiply with a fixed sequence of adds and subtracts which have the same effect. For instance, multiply by 5 could be achieved using a single instruction:

    ```
    ADD    Rd, Rm, Rm, LSL #2              ; Rd = Rm + (Rm * 4) = Rm * 5
    ```
    (LSL #2 - left shift the register 2 places to the left (equivalent to multiplying by 4))

    This ADD version is better than the MUL version below:
    ```
     MOV    Rs, #5
      MUL    Rd, Rm, Rs
    ```

    The 'cost' of the general multiply includes the instructions needed to load the constant into a register (up to 4 may be needed, or an LDR from a literal pool) as well as the multiply itself.

    Consider multiply by 105: 105 can be written as 105 = 128 - 16 + 2 + 1. To perform this we can do as below
    ```
    ADD    Rd, Rm, Rm, LSL #1                  ; Rd = Rm*3
    SUB    Rd, Rd, Rm, LSL #4                  ; Rd = Rm*3 - Rm*16
    ADD    Rd, Rd, Rm, LSL #7                  ; Rd = Rm*3 - Rm*16 + Rm*128
    ```

    Or, decomposing differently:

    105 = 15 * 7
        = (16 - 1) * (8 - 1)

```
RSB    Rt, Rm, Rm, LSL #4                    ; Rt = Rm*15 (tmp reg)
RSB    Rd, Rt, Rt, LSL #3                   ; Rd = Rt*7 = Rm*105
```

Second is optimal. So find an optimal way to solve.

5. Initiate a simple stack. Use the Store(STM) instruction to push 4 registers into the stack and Load(LDM) instruction to pop the registers out of the stack. Try using all the 4 types of stack operations viz Full Ascending(FA), Full Descending(FD), Empty Ascending(EA), Empty Descending(ED) and see the differences.
6. Sort a sequence of 5 numbers in ascending order and store it in memory.