Homework 5 (100 points)

Due Date*: 10:00am 03/07/2018, Cutoff Deadline: 10:00am 03/10/2018**

** **Submission will NOT be accepted after the cutoff deadline**

Submission: The .s file on Blackboard under Homework 5 (NO email submission please!)

Warning: (1) The grading of an ARM program is based on the testing results. An ARM program that cannot be assembled will receive up to 5% of the points!!! (2) Once the submitted work is graded, no more submission will be accepted or re-graded.

Write an ARM program to read and convert a 32-bit 2's complement into a signed **decimal** string

- in the data area,
  - Declare and initialize a ***null-terminated string*** with a label of **HexStr** and an initial value of your choice (each byte is the ASCII of a Hex symbol in this 32-bit 2's complement number). For example, the string could be "A8F", 0.
  - declare and initialize a signed word as 0, label this word as **TwosComp**
  - reserved a chunk of zeroed memory with a length of 12 bytes, label this chunk of memory as **DecStr** (why a length of 12 bytes is enough?)
  - reserved a chunk of zeroed memory with a length of 11 bytes, label this chunk of memory as **RvsDecStr** (why a length of 11 bytes is enough?)
  - Store each one of the above four labels as a word with an address label, e.g., "adrHexStr DCD HexStr" for HexStr, and **EXPORT** each address label, e.g., "EXPORT adrHexStr".
- in the main program,
  - read the ASCII's of up to eight symbols of a Hex number (e.g., 1A2B3D4D or FFF4B3FA) symbol by symbol from the memory at HexStr, and convert this list of symbols in ASCII into a 32-bit two's complement number. You may assume that this Hex number will be zero-extended if less than eight symbols are read, i.e., the missing leading symbols are zeros (e.g., A79 means 00000A79).
  - store this 32-bit 2's complement number at TwosComp.
  - convert the 32-bit signed number at TwosComp to a NULL terminated ASCII string, each byte is the ASCII of the **minus sign** or a **digit** in this number. A minus sign, '–', needs to be placed at the beginning if this number is negative. E.g., if [TwosComp] = 0x1A2B3C4D, then it should be converted into "439041101" ; if [TwosComp] = 0xFFF4B3FA, then it should be converted into "–740358" .
    - Hint: when you keep dividing the **absolute value of this number** by 10, the list of remainders that you obtain are the digits whose ASCIIs are going to be stored at DecStr. However, the first remainder is the Least Significant Digit and the last remainder is the Most Significant Digit!
    - Therefore, you might want to store the ASCIIs of the digits you obtained through division to RvsDecStr first, then reversely store them to the memory at DecStr.
  - write a your own **subroutine** to divide a **positive** 32-bit number by **10** and put the **quotient** and the **remainder** in two registers, respectively, as the output parameters.
  - call this subroutine for division-by-10 while converting the signed number into its decimal equivalent. You are NOT allowed to call **the division subroutine** that is provided by the emulator.
  - Store this decimal string as a NULL-terminated string to memory labeled as DecStr.