Assignment 5 (3%)
Integer to ASCII (itoa)

Deadline: Friday 12 November 2021 at 23:59 on Submitty

Working individually, complete the assignment below. Submit your solution to Submitty (`https://submit.scss.tcd.ie`). By submitting your solution, you are confirming that you have familiarised yourself with College's policy on plagiarism (`https://libguides.tcd.ie/plagiarism`).

Your mark will be the auto-graded mark assigned by Submitty (10 marks).

You are allowed to submit five attempts for the assignment without penalty. Subsequent attempts will attract a 1 mark penalty each, up to a maximum penalty of 5 marks.

Submitty will allow you eight "late days" over the full semester. This means, for example, you can submit one assignment late by eight days or eight assignments late by one day each, without penalty. Once your "late days" are used up, you will receive zero marks for any late submissions.

Instructions

Design and write an ARM Assembly Language program that will create an ASCII string in memory representing the signed value in register R1 in decimal form.

Begin storing your string in memory at the memory address that is in R0.

For example, if R1 contains the value $+45$, your program should store the ASCII characters '+', '4', '5' in memory beginning at the address in R0. Alternatively, if R1 contains the value -365, your program should store the ASCII characters '-', '3', '6', '5' in memory beginning at the address in R0.

If the value in R1 is 0, your program should store the ASCII character '0' in memory beginning at the address in R0 and **should not prefix it with a sign**.

Hint 1: Begin with a pseudo-code solution which you can then translate into ARM Assembly Language.

Hint 2: You may find it easier to begin by writing a program that only works for positive numbers. Taking this approach, a working program would be able to pass tests worth 7 out of the 10 marks available. You can then extend your program to work for negative numbers and zero for the remaining 3 marks.

Hint 3: You may wish to use the ARM `UDIV` instruction. For example, `UDIV R0, R1, R2` divides R1 by R2 and stores the whole part (quotient) of the result in R0. The remainder from the division is not available but you can compute it with some basic arithmetic (see Assignment 4).

Hint 4: One of the challenges in this assignment is to produce the digits in left-to-right order. One approach is to begin by finding the largest power of 10 that is smaller than the value your program is converting. You can then divide by decreasing powers of 10, taking the quotient as

your next digit and the remainder as your next dividend. Other approaches can also work.

Hint 5: Don't forget to NULL-terminate your string in memory. (i.e. Store the value zero after the last ASCII character.)