# Assignment 4 (5%)
# Bit Manipulation

Deadline: Friday, 1 December 2023 at 23:59 on Submitty

**Working individually, complete the assignment below. Submit your solution to Submitty (`https://submit.scss.tcd.ie`). Your mark will be the auto-graded mark assigned by Submitty (100 marks).**

You are allowed to submit **six attempts** for the assignment without penalty. Subsequent attempts will attract a 5 mark penalty each, up to a maximum penalty of 30 marks.

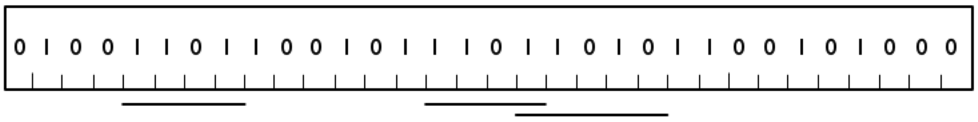
Submitty will allow you **eight "late days"** for CSU11021 assignments. For example, you can submit one assignment late by eight days or four assignments late by two days each, without penalty. Once your "late days" are used up, you will receive zero marks for any late submissions. Note that even 1 second after the deadline counts as one full late day. Late days do not apply to exercises that are not for credit.

**By submitting your solution, you are confirming that you have familiarised yourself with College's policy on academic integrity (`https://libguides.tcd.ie/academic-integrity`).**

## Instructions

Write an ARM Assembly Language program that will count the number of occurrences of the bit pattern 1101 in a block of data in memory. The block of data begins at the address in R1. The length of the block of data, measured in bytes, is in R2.

The occurrences may overlap each other. For example, in the block of data below, there are three occurrences, two of which overlap.



The occurrences may span byte boundaries in memory (and half-word and word boundaries).

60 marks will be based on tests for correct functionality. The remaining 40 marks will be based on the performance of your program, measured in processor cycles.

### Hints:

Submitty will test the simple cases first, so start with a simple version of the program and gradually extend it to handle more complexity.

(i) Start with a program that can count the number of occurrences of 1101 in a single byte from memory.

(ii) Make sure your program handles cases where occurrences of 1101 overlap each other.

(iii) Extend your program to handle more than one byte but ignore the potential for occurrences to span byte (or half-word or word) boundaries.

(iv) Work on improving the performance of your program (if you are not already getting full performance marks). Note that loading a byte (`LDRB`) uses the same number of cycles as loading a word (`LDR`) so it is more efficient to load four bytes at a time by loading a word. (But be careful! The number of bytes in the block of data might not be evenly divisible by 4.)

(v) Finally, extend your program again to handle the case where the sequence might overlap a byte boundary (or half-word or word) boundary. This is likely to be the hardest part of the assignment. You may need to consider endianness and may wish to use the `REV` instruction to reverse the ordering of bytes in a register.

Performance calculation assumptions:

(i) Each instruction execution takes at least one cycle.

(ii) Divisions (e.g. UDIV) take an extra 3 cycles.

(iii) Branches that are taken take an extra 2 cycles.

(iv) Loads and stores take one extra cycle regardless of size (word, halfword, byte) but

(v) loads and stores of words or halfwords that span a word boundary take two extra cycles.