# CIS11 Lab 11: Random Number Generator

## Problem Statement

• Generate random numbers using a **Linear Congruential Random Number Generator (LCRNG).**

## Inputs and Outputs

The seed, which is an integer in the range **1 to 32766**, is found at location **x3100**. When the program is executed, **20 random numbers** in the interval **1 to 215−2** are generated and displayed.

## Linear Congruential Random Number Generators

A LCRNG is defined by the recurrence equation:

**Xn 🡨 A\*XN-1 + C mod M**

The multiplicative constant A, the constant C, and modulus M are integers that are chosen and fixed.

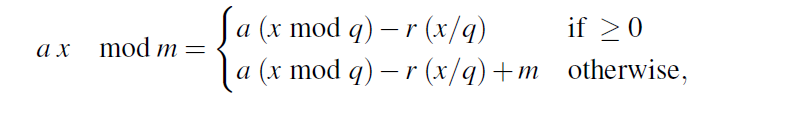
Given the seed x0, a random number sequence is generated: x1,x2,x3, . . . , with the xi’s being in the range 0 to m−1. Eventually the sequence will repeat itself. In most cases, it is desirable that the period of repetition is as long as possible.

Using the subroutines MULT and DIV, used in earlier labs, one can write a program in LC-3 to generate random numbers based on equation

There is, however, the possibility that intermediate operations, such as a xn−1, cause an overflow. In the case where c=0, to avoid overflow we use Schrage’s method1. In this method, the recurrence is

**Xn 🡨 A\*XN-1 mod M**

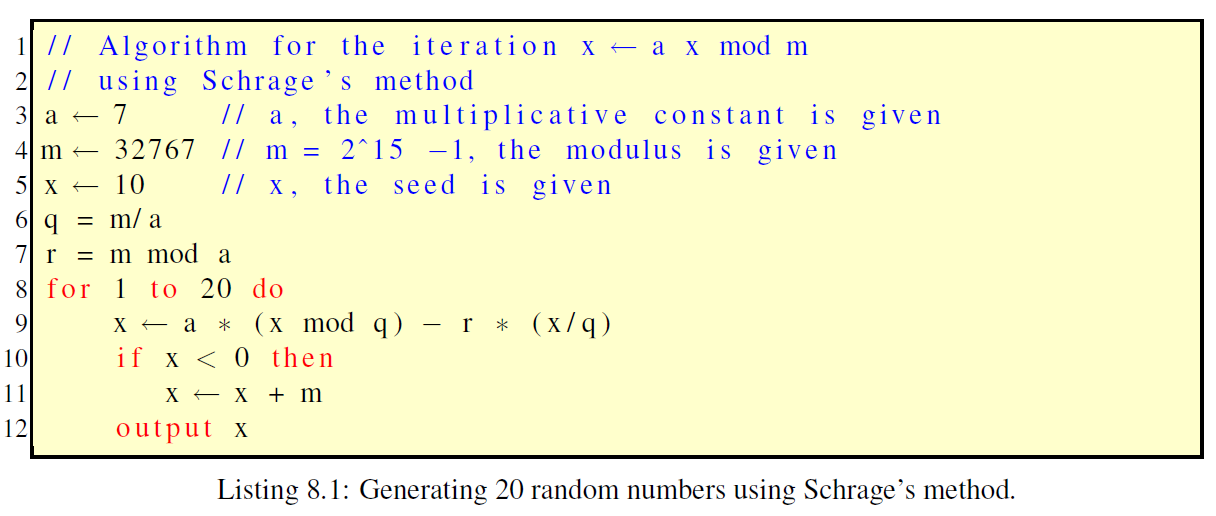
and multiplication a x is performed in the following fashion (equation 8.3):



Where

**q = m/a, r = m mod a**

As always, “/” denotes integer division. To ensure no overflow while performing the computations in equation (8.3), multiplier **a** and **m**odulus m must be chosen so that 0 ≤ r < q.



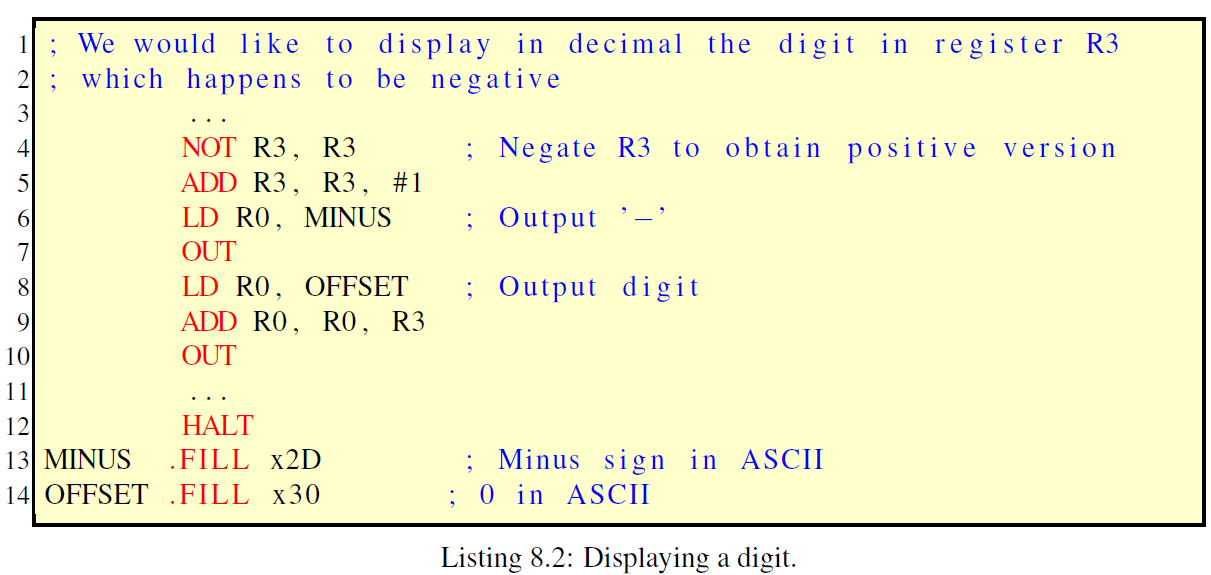
For two’s complement 16-bit arithmetic, which is the LC-3 case, the largest possible m is 215−1.

Using this value for m, to produce a maximal non-repeating sequence (all integers in the range 1 to 215 −2, will be generated before the sequence will repeat itself) of random numbers one can choose a = 7. The seed x0 should never be 0; it should be any number from 1 to 215−2 = 32766.

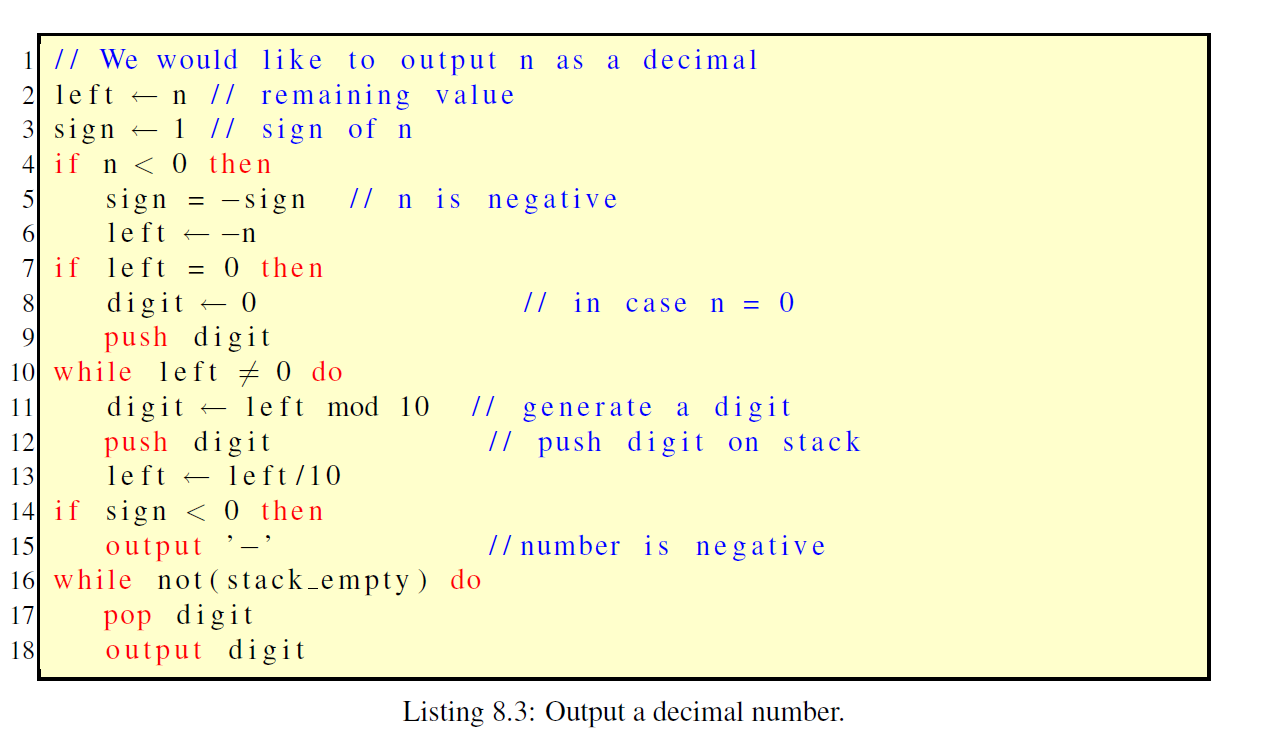
Your program should implement equation

**Xn 🡨 A\*XN-1 mod M**

The assembly command OUT, which is shorthand for TRAP x21, outputs the single ASCII character found in the 8 least significant bits of R0. We can use OUT

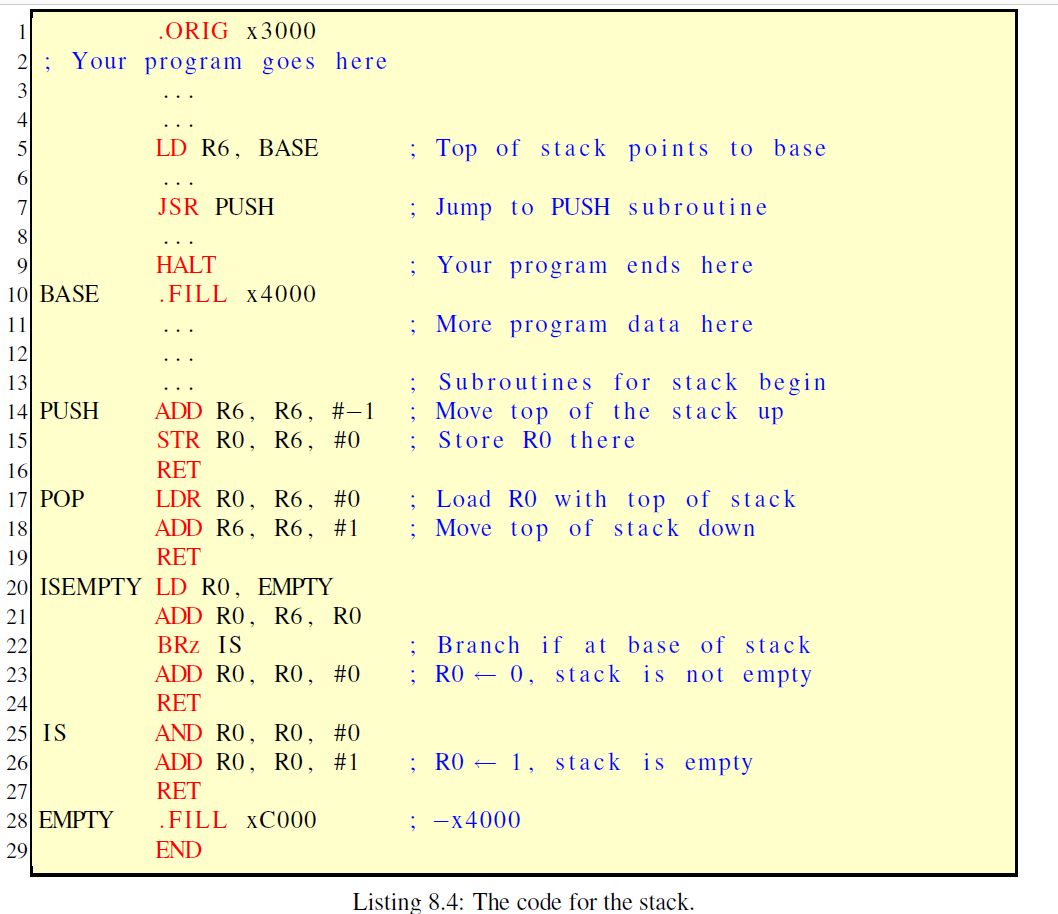


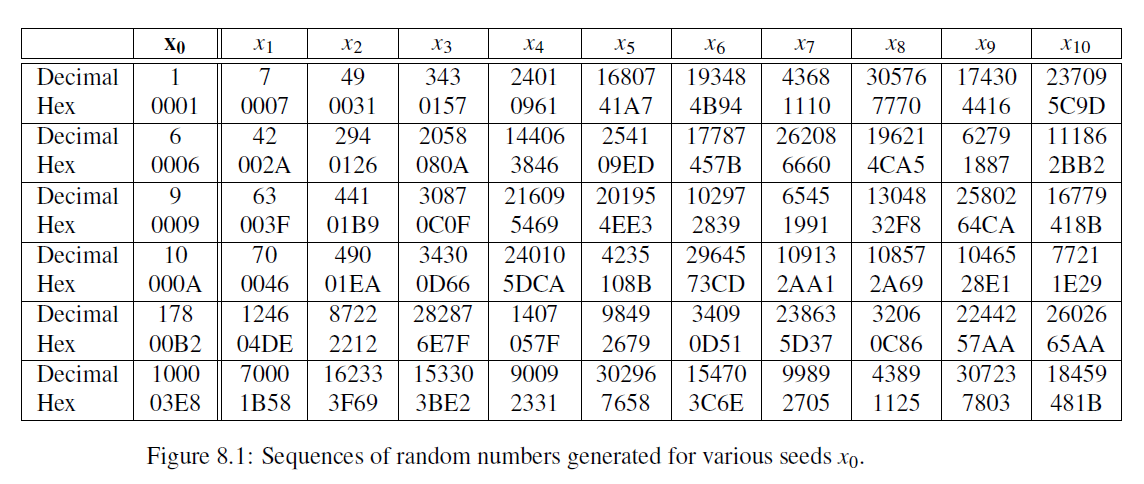
Therefore, to output the decimal digits of a number one by one. We can obtain the digits by successively applying the mod 10 on the number and truncating, until we obtain 0. This produces the digits from right to left. For example if the number we would like to output is x219 = **537**, by applying the above procedure we obtain the digits in this order: **7,3, 5**. Thus, we have to output them in reverse order of their generation. For this purpose we can use a stack, with operations **PUSH** and .



The stack that is described here is a rudimentary one. It is intended for this problem only. There are three operations, i.e. subroutines that involve the stack: **PUSH, POP**, and **ISEMPTY**. PUSH pushes the contents of register R0 on the stack, POP pops the top of the stack in register R0, and ISEMPTY returns 1 in R0 if the stack is empty and 0 if the stack is non-empty. Register **R6** points to the top of the stack. The following have to be borne in mind when writing your program:

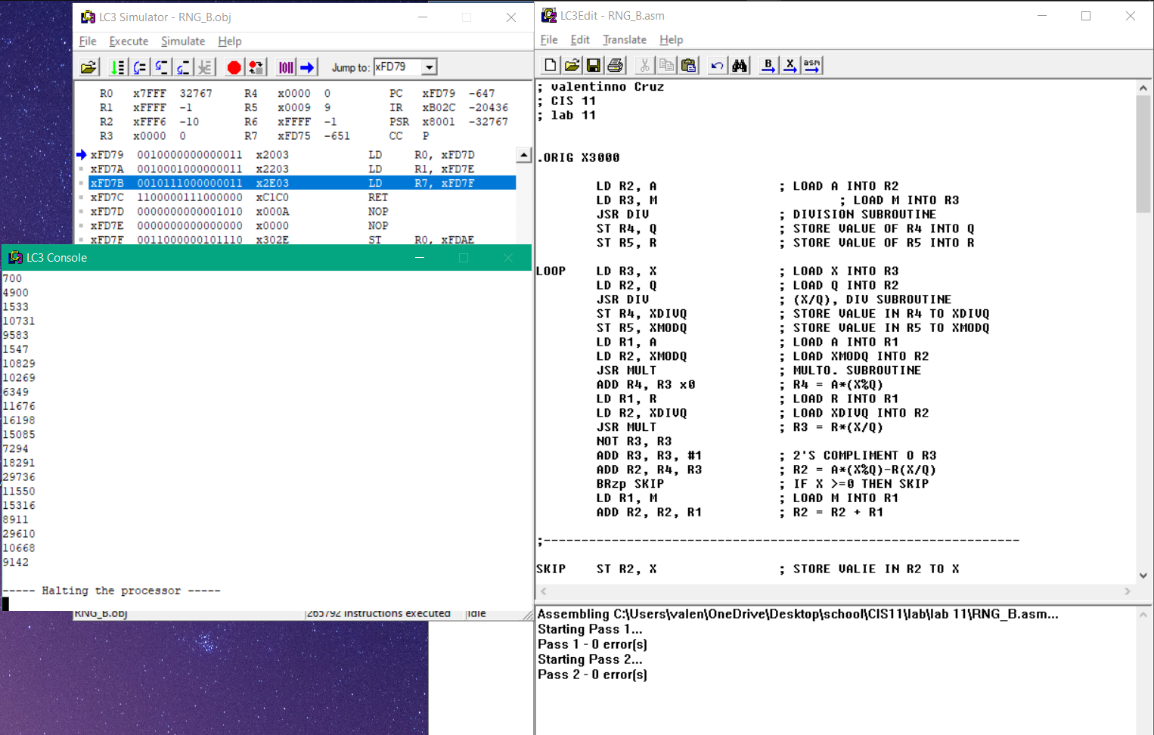
* **R6** should be initialized to **x4000**, the base of the stack, and not be overwritten while manipulating the stack.
* **R7** will be used (implicitly) to store the return address when calling a subroutine.
* Always **ISEMPTY** should be called before proceeding to call **POP**, to check whether the stack is empty. If empty, **POP** should not be called.





## Tasks

1. **Create an LC-3 Random Number Generator using the recurrence equation and the provided lab information.**
2. **Take a screen capture of: code (in Editor), successful running program in Simulator, and output in Console.**
3. **Test the program using a = 7, m = 32767 in equation and starting with various seeds x0, the first 10 random numbers generated in each case are listed in figure 8.1.**
4. **How would you adjust the program to display more randomly generated numbers?**
5. **Pertain to the program, explain how subroutines are used for stack operations.**

****

**C) changing the value of LC will generate as many numbers as we want.**

**D) the stack is calling upon the subroutines. To help navigate the flow of data the program is doing. Itll call onto a number of subroutines in the main loop. Then RET op will pop the calling address from the stack, and also push the return value if needed onto the stack.**