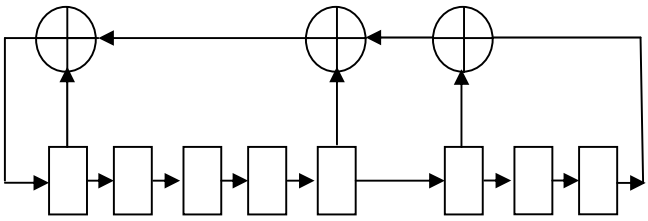
COE 205, Term 061

Computer Organization & Assembly Programming
Programming Assignment# 3

Due date: Wednesday, Dec. 13, 2006

You are required to write an 8086 assembly program to implement a pseudo random generator using Liner Feedback Shift Register (LFSR). An example of an 8-bit LFSR is shown below:



Two important characteristics of an LFSR are the Feedback Polynomial, which determines the FFs that are XORed to compute the shifted bit, and the seed which determines the initial content of the FFs. Depending on the Feedback polynomial, the LFSR can generate a maximal-length sequence without repetition, or it may not. The seed can be any number other than 0.
The 8-bit LFSR shown above is a maximal-length i.e. it is guaranteed to generate a random sequence in the range from 1 to 255 before it repeats again.
The Feedback polynomial for the above LFSR can be represented as 10001101. Note that 1 indicates that there is feedback connection, while 0 indicates that there is no feedback connection.

(i) Write a procedure, **RAND8**, that implements an 8-bit pseudo random generator. The procedure should be given the Feedback polynomial, and the seed, and it should generate the next random number.

(ii) Write a program that generates all 8-bit primitive polynomials. Note that there are 255 different polynomials. Your program should print the number of primitive polynomials and found and print the primitive polynomials themselves.

(iii) Write a procedure, **ToOneDigit,** that converts a multiple digit 8-bit number to a single digit number. This is achieved by adding the individual digits of this number to get another integer. The same process is repeated to the obtained integer. This process is repeated until the number becomes a single digit. For exampe: 219=2+1+9=12=1+2=3. Thus, the procedure converts 219 to 3.

(iv) Write a procedure, **ToOneCharacter**, that converts a string of characters to a single character by XORing the ASCII code of these characters. If the result is 0, then make it 1, otherwise keep it as is.

(v) Write a program that asks the user to enter a password, which is a string of maximum 30 charcaters. Use procedure ToOneCharacter to convert the password into a single character to be used as the seed of the random number generator. Then, ask the user to enter a string of characters. Then, encrypt the string using RAND8 and ToOneDigit procedures as follows. Store all the primitive polynomials in a table called **Primitives**. For each character, use the procedure RAND8 to generate a random number using the current seed and the next primitive polynomial in the table Primitives. Then, use procedure ToOneDigit to convert the generated random number into a single digit number, N. The next available character in the string is encrypted by XORing the least significant 4-bits of the ASCII code of the character with N. For example, assume the character to be encrypted is 'A'=41H and the random number is 123d. Then, the random number is converted to a single digit number N=6d. Then, the character is encrypted by XORing 1 XOR 6 and the encrypted character will be 47h='G'. To decrypt the character, the encrypted character 47H='G', will be XORed with the same corresponding single digit number N=6 obtained from the same random number used for encryption i.e. 123 and this will generate the original character 41H='A', as 1 XOR 7 =6. As an example show the encryption of the string **This is the last Assignment!!**. Then, rerun your program giving it the encrypted string and it should correctly decrypt it to **This is the last Assignment!!**. Try this with a seed of 10101010.

The solution should be well organized and your programs should be well documented. Submit a soft copy of your solution in a zip file. The soft copy should include a Readme file indicating the file names containing the solution and whether it works or not. The Readme file should also contain your name and ID. Submit both source code files (i.e. .asm) and the executable files (i.e. .exe).