COE 301/ICS 233, Term 171

Computer Architecture & Assembly Language

HW# 3

Q.1. Write a MIPS assembly program that does the following:

(i) Ask the user to enter the number of rows R and read it.

(ii) Ask the user to enter the number of columns C and read it.

(iii) Ask the user to enter a two-dimensional array of RxC characters. Elements of a single row should be separated by a single space and each row is read in a new line.

(iv) Print the entered array in a new line printing also its entered dimensions.

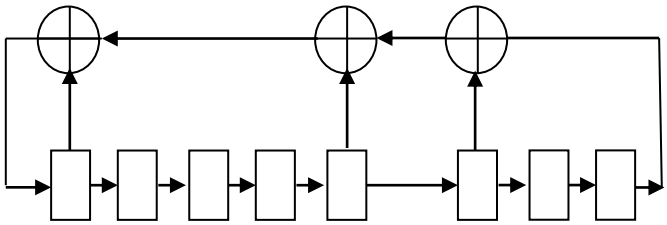
(v) Ask the user to enter two row numbers

(vi) Exchange the two entered rows and print the array after the exchange.

A sample execution of the program is shown below:

Enter number of rows in the array: 3
Enter number of columns in the array: 5
Enter an array of 3x5 characters:
0 1 2 3 4
5 6 7 8 9
a b c d e
The entered 3x5 array is:
0 1 2 3 4
5 6 7 8 9
a b c d e
Enter a row number: 0
Enter another row number: 2
The array after exchanging Row 0 and Row 2 is:
a b c d e
5 6 7 8 9
0 1 2 3 4

Q.2. You are required to write a MIPS 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 flip-flops (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 **READB** to read a binary number and store it in $v0. The procedure should report an error message and ask the user to reenter the number if the decimal value of the entered number is 0. Also, the user does not have to enter the whole 8-bits. If he enters less than 8 bits and hits return the remaining most significant bits should be assumed 0. Also, if any digit entered is other than a binary digit, an error message should be reported.

(ii) Write a procedure, **RAND8**, that implements an 8-bit pseudo random generator. The procedure should be given the Feedback polynomial, and the seed as parameters in $a0 and $a1 registers and it should generate the next random number in $v0.

(iii) Ask the user to enter an 8-bit feedback ploynomial and an 8-bit seed in binary. Use the procedure READB for this purbose. Then, ask the user to enter a string of characters. Then, encrypt the string using RAND8 as follows. Each character is encrypted by XORing the least significant 4-bits of the ASCII code of the character with the least significant 4 bits and the most significant 4-bits of the generated random number . For example, assume the character to be encrypted is 'A'=0x41 and the random number is 0xA1. Then, the encrypted character will have the ASCII code 0x4A='J'. To decrypt the character, the decrypted character 0x4A='J', will be XORed with the same corresponding random number used for encryption i.e. 0xA1 and this will generate the original character 0x41='A'. As an example, show the encryption of the string **"This is an interesting assignment"**. Then, rerun your program giving it the encrypted string and it should correctly decrypt it to "**This is an interesting assignment**". Encrypt this with the feedback polynomial 10001101 and a seed of 00001111.