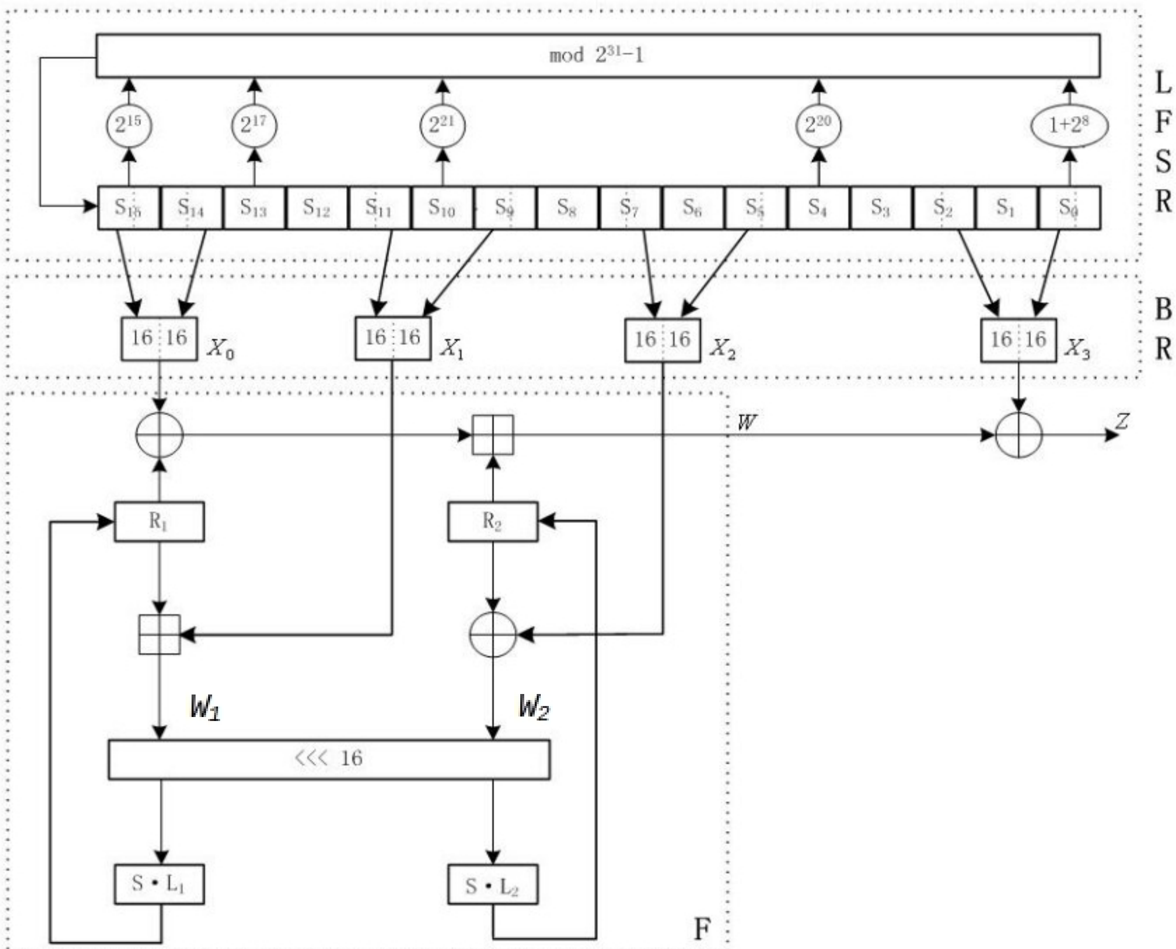
Lab: ZUC specification in Cryptol

ZUC is a word-oriented stream cipher intended for cell phone encryption and decryption. It takes a 128-bit initial key and a 128-bit initial vector as input and outputs a keystream of 32-bit words. The Figure below shows the schematic diagram from which ZUC is derived.



The section labeled `LFSR` shows a Linear Feedback Shift Register. This register has 16 stages labeled s_0 to s_{15}. Each stage is 31 bits wide. For all stages, a clock moves data from input to output, simultaneously, from s_i to s_{i-1} for `i` from 15 to 1. Input to s_{15} comes from the output of a function of values from stages s_{15}, s_{13}, s_{10}, s_4, and s_0. This function is defined below during an exercise. The section labeled `BR` shows the Bit Reorganization registers that contribute to inputs in the Mangler function `F` and to the output keystream `Z`. There are 4 such registers, X_0, X_1, X_2, X_3, taking values from half (16 bits) of stages s_{15}, s_{14}, s_{11}, s_9, s_7, s_5,

S_2, and S_0. The Mangler function F takes input from the BR registers and outputs a 32 bit wide stream that is xored with BR register X_3 to provide the keystream output Z.

The execution of ZUC has two stages: initialization stage and working stage. In the first stage, a key/IV initialization is performed, i.e., the cipher is clocked without producing output. The second stage is a working stage. In this stage, with every clock pulse, a 32-bit word of output is produced.

Exercise 1:
Write a function `plus (a,b)` that takes two 31 bit integers `a` and `b` as input, adds them `mod` $(2^{31}-1)$, and returns the resulting 31 bit integer.

Exercise 2:
Write a function `add xs` that takes a sequence `xs` of 31 bit integers as input and adds them all `mod` $(2^{31}-1)$. Use the result of Exercise 1 in a comprehension.

Exercise 3:
Write a function `v ss` that takes as input a sequence `ss` of 16, 31 bit integers, and returns

$$(S_{15}<<<_{31}15)+(S_{13}<<<_{31}17)+(S_{10}<<<_{31}21)+(S_4<<<_{31}20)+(S_0<<<_{31}8)+S_0 \bmod (2^{31}-1)$$

Use the result of Exercise 2 on a comprehension.

Exercise 4:
In the Initialization mode, the LFSR receives a 31-bit input word u, which is obtained by removing the rightmost bit from the 32-bit output W of the nonlinear function F, i.e., u=W>>1. More specifically, according to the ZUC specification, the initialization mode works as follows:

```
LFSRWithInitialisationMode (u) {
   v=(S15<<<3115)+(S13<<<3117)+(S10<<<3121)+(S4<<<3120)+(S0<<<318)+S0 mod (2^31-1)
  S16 = (v+u) mod (2^31-1)
  if S16 == 0, then set S16 = 2^31-1;
  (S1,S2, …,S15,S16) -> (S0,S1, …,S14,S15).
}
```

Write `LFSRWithInitializationMode` as a Cryptol function using the results of Exercises 1 to 3. You will need to add `ss`, representing the 16 stage sequence of the LFSR, to the argument list of the function. Note that the addition of `v` and `u` is a `mod` $(2^{31}-1)$ addition.

Exercise 5:
In Work mode LFSR operates like this according to the specification:

```
LFSRWithWorkMode () {
 s16=(S15<<<3115)+(S13<<<3117)+(S10<<<3121)+(S4<<<3120)+(S0<<<318)+S0 mod (2^31-1)
 if s16==0, then set s16=2^31-1;
 (s1,s2, …,s15,s16) -> (s0,s1, …,s14,s15).
}
```

Write `LFSRWithWorkMode` as a Cryptol function using the results of Exercises 1 to 3. You will need to add `ss` to the argument list of the function.

Exercise 6:

The `BR` section extracts 128 bits from the `LFSR` stages and forms 4, 32-bit words, where the first three words are used by the nonlinear function `F` in the bottom layer, and the last word is involved in producing the keystream. According to the `ZUC` specification this may be expressed as:

```
Bitreorganization () {
  X0 = S15H || S14L;
  X1 = S11L || S9H;
  X2 = S7L || S5H;
  X3 = S2L || S0H.
}
```

where the `||` operation is concatenation, and the S_i are 31-bit integers, so $S_i H$ means bits 30...15 and not 31...16 of S_i, for $0 \le i \le 15$. Write the `Bitreorganization` function in Cryptol. You will need to add `ss` to the argument list of the function.

Exercise 7:

The 32×32 S-box S is composed of 4 juxtaposed 8×8 S-boxes, i.e., `S=(S0,S1,S2,S3)`, where `S0=S2`, `S1=S3`. The definitions of `S0` and `S1` can be found below as `S0Box` and `S1Box`, respectively.

Let `x` be an 8-bit input to `S0` (or `S1`). Write `x` into two hexadecimal digits as `x=h||l`, then the entry at the intersection of the `h`th row and the `l`th column in `S0Box` or `S1Box` is the output of `S0` (or `S1`). So, define function `S X`, where `X` is a 32 bit integer, to be the concatenation of the `SBox` lookups for the 4 bytes of `X`. That is, the high order byte lookup is in `S0Box`, the next is in `S1Box`, then `S0Box` and `S1Box`. Write this function, and all supporting functions, in Cryptol.

```
S0Box =
  [0x3E, 0x72, 0x5B, 0x47, 0xCA, 0xE0, 0x00, 0x33, 0x04, 0xD1, 0x54,
   0x98, 0x09, 0xB9, 0x6D, 0xCB, 0x7B, 0x1B, 0xF9, 0x32, 0xAF, 0x9D,
   0x6A, 0xA5, 0xB8, 0x2D, 0xFC, 0x1D, 0x08, 0x53, 0x03, 0x90, 0x4D,
   0x4E, 0x84, 0x99, 0xE4, 0xCE, 0xD9, 0x91, 0xDD, 0xB6, 0x85, 0x48,
   0x8B, 0x29, 0x6E, 0xAC, 0xCD, 0xC1, 0xF8, 0x1E, 0x73, 0x43, 0x69,
   0xC6, 0xB5, 0xBD, 0xFD, 0x39, 0x63, 0x20, 0xD4, 0x38, 0x76, 0x7D,
   0xB2, 0xA7, 0xCF, 0xED, 0x57, 0xC5, 0xF3, 0x2C, 0xBB, 0x14, 0x21,
   0x06, 0x55, 0x9B, 0xE3, 0xEF, 0x5E, 0x31, 0x4F, 0x7F, 0x5A, 0xA4,
   0x0D, 0x82, 0x51, 0x49, 0x5F, 0xBA, 0x58, 0x1C, 0x4A, 0x16, 0xD5,
   0x17, 0xA8, 0x92, 0x24, 0x1F, 0x8C, 0xFF, 0xD8, 0xAE, 0x2E, 0x01,
   0xD3, 0xAD, 0x3B, 0x4B, 0xDA, 0x46, 0xEB, 0xC9, 0xDE, 0x9A, 0x8F,
   0x87, 0xD7, 0x3A, 0x80, 0x6F, 0x2F, 0xC8, 0xB1, 0xB4, 0x37, 0xF7,
   0x0A, 0x22, 0x13, 0x28, 0x7C, 0xCC, 0x3C, 0x89, 0xC7, 0xC3, 0x96,
   0x56, 0x07, 0xBF, 0x7E, 0xF0, 0x0B, 0x2B, 0x97, 0x52, 0x35, 0x41,
   0x79, 0x61, 0xA6, 0x4C, 0x10, 0xFE, 0xBC, 0x26, 0x95, 0x88, 0x8A,
   0xB0, 0xA3, 0xFB, 0xC0, 0x18, 0x94, 0xF2, 0xE1, 0xE5, 0xE9, 0x5D,
   0xD0, 0xDC, 0x11, 0x66, 0x64, 0x5C, 0xEC, 0x59, 0x42, 0x75, 0x12,
   0xF5, 0x74, 0x9C, 0xAA, 0x23, 0x0E, 0x86, 0xAB, 0xBE, 0x2A, 0x02,
   0xE7, 0x67, 0xE6, 0x44, 0xA2, 0x6C, 0xC2, 0x93, 0x9F, 0xF1, 0xF6,
   0xFA, 0x36, 0xD2, 0x50, 0x68, 0x9E, 0x62, 0x71, 0x15, 0x3D, 0xD6,
   0x40, 0xC4, 0xE2, 0x0F, 0x8E, 0x83, 0x77, 0x6B, 0x25, 0x05, 0x3F,
   0x0C, 0x30, 0xEA, 0x70, 0xB7, 0xA1, 0xE8, 0xA9, 0x65, 0x8D, 0x27,
   0x1A, 0xDB, 0x81, 0xB3, 0xA0, 0xF4, 0x45, 0x7A, 0x19, 0xDF, 0xEE,
   0x78, 0x34, 0x60]
```

```
S1Box =
  [0x55, 0xC2, 0x63, 0x71, 0x3B, 0xC8, 0x47, 0x86, 0x9F, 0x3C, 0xDA,
   0x5B, 0x29, 0xAA, 0xFD, 0x77, 0x8C, 0xC5, 0x94, 0x0C, 0xA6, 0x1A,
   0x13, 0x00, 0xE3, 0xA8, 0x16, 0x72, 0x40, 0xF9, 0xF8, 0x42, 0x44,
   0x26, 0x68, 0x96, 0x81, 0xD9, 0x45, 0x3E, 0x10, 0x76, 0xC6, 0xA7,
   0x8B, 0x39, 0x43, 0xE1, 0x3A, 0xB5, 0x56, 0x2A, 0xC0, 0x6D, 0xB3,
   0x05, 0x22, 0x66, 0xBF, 0xDC, 0x0B, 0xFA, 0x62, 0x48, 0xDD, 0x20,
   0x11, 0x06, 0x36, 0xC9, 0xC1, 0xCF, 0xF6, 0x27, 0x52, 0xBB, 0x69,
   0xF5, 0xD4, 0x87, 0x7F, 0x84, 0x4C, 0xD2, 0x9C, 0x57, 0xA4, 0xBC,
   0x4F, 0x9A, 0xDF, 0xFE, 0xD6, 0x8D, 0x7A, 0xEB, 0x2B, 0x53, 0xD8,
   0x5C, 0xA1, 0x14, 0x17, 0xFB, 0x23, 0xD5, 0x7D, 0x30, 0x67, 0x73,
   0x08, 0x09, 0xEE, 0xB7, 0x70, 0x3F, 0x61, 0xB2, 0x19, 0x8E, 0x4E,
   0xE5, 0x4B, 0x93, 0x8F, 0x5D, 0xDB, 0xA9, 0xAD, 0xF1, 0xAE, 0x2E,
   0xCB, 0x0D, 0xFC, 0xF4, 0x2D, 0x46, 0x6E, 0x1D, 0x97, 0xE8, 0xD1,
   0xE9, 0x4D, 0x37, 0xA5, 0x75, 0x5E, 0x83, 0x9E, 0xAB, 0x82, 0x9D,
   0xB9, 0x1C, 0xE0, 0xCD, 0x49, 0x89, 0x01, 0xB6, 0xBD, 0x58, 0x24,
   0xA2, 0x5F, 0x38, 0x78, 0x99, 0x15, 0x90, 0x50, 0xB8, 0x95, 0xE4,
   0xD0, 0x91, 0xC7, 0xCE, 0xED, 0x0F, 0xB4, 0x6F, 0xA0, 0xCC, 0xF0,
   0x02, 0x4A, 0x79, 0xC3, 0xDE, 0xA3, 0xEF, 0xEA, 0x51, 0xE6, 0x6B,
   0x18, 0xEC, 0x1B, 0x2C, 0x80, 0xF7, 0x74, 0xE7, 0xFF, 0x21, 0x5A,
   0x6A, 0x54, 0x1E, 0x41, 0x31, 0x92, 0x35, 0xC4, 0x33, 0x07, 0x0A,
   0xBA, 0x7E, 0x0E, 0x34, 0x88, 0xB1, 0x98, 0x7C, 0xF3, 0x3D, 0x60,
   0x6C, 0x7B, 0xCA, 0xD3, 0x1F, 0x32, 0x65, 0x04, 0x28, 0x64, 0xBE,
   0x85, 0x9B, 0x2F, 0x59, 0x8A, 0xD7, 0xB0, 0x25, 0xAC, 0xAF, 0x12,
   0x03, 0xE2, 0xF2]
```

Exercise 8:
Cryptol specification of the nonlinear function F is a little tricky. There are two linear transformations L1 and L2 that are applied to an S input. These are:

$$L1(X)=X \oplus (X <<<_{32} 2) \oplus (X <<<_{32} 10) \oplus (X <<<_{32} 18) \oplus (X <<<_{32} 24),$$
$$L2(X)=X \oplus (X <<<_{32} 8) \oplus (X <<<_{32} 14) \oplus (X <<<_{32} 22) \oplus (X <<<_{32} 30).$$

Write Cryptol functions for these.

Exercise 9:
Build `F` in steps.

a) The output of `F` is `W` and is written as $(X_0 \oplus R_1) \boxplus R_2$ in the specification where $\oplus$ is exclusive or and $\boxplus$ is mod 2^{32} addition. R_1 and R_2 are shown in the Figure above. Write an expression for this.
b) See W_1 and W_2 in the Figure above and Section 3.4 in the ZUC specification. Write expressions for W_1 and W_2.
c) W_1 and W_2 get split into halves W_1L, W_1H, W_2L, W_2H. Then
$R_1=S(L_1(W_1L||W_2H))$ and $R_2=S(L_2(W_2L||W_1H))$
Write expressions to get the halves and use that to write R_1 and R_2.
d) Write the full expression for nonlinear function `F`. Notice that `F` takes $[X_0, X_1, X_2]$ and the sequence $[R_1,R_2]$ as input and outputs (`W, [R1′,R2′]`) where `R1′` and `R2′` are obtained in c) above.

Exercise 10:
Write a function that completes the key loading operation stated in Section 3.5. Call the function `LoadKey`, have it take arguments `key:[128]` and initialization vector `iv:[128]` and have it output values held in the 16 stages of the `LFSR`.