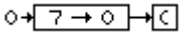
SHIFT or ROTATE an array of integers

On occasion I have had the fairly niche requirement to shift an integer variable (64 bits) left or right but preserve whatever fell off the end. A larger version of the common shift/rotate instructions of CPUs - many of which have far more complicated types.

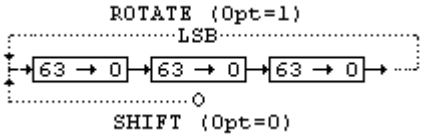
My requirement was fairly simple, shift an integer and where the least significant bit (LSB), say, would normally be lost, preserve it and move it into the most significant bit (MSB) of the next.

Consider the action of the Z80 instruction SRL (Shift Right Logical)



The byte in the register is bit-wise shifted right, a zero is shifted into the MSB and the LSB is moved into the carry flag. At its simplest I needed this functionality and its counter part the Shift Left Logical. Using combinations of these instructions it was possible to rotate/shift huge arrays of bytes using the carry to perpetuate the orphaned bit into the relevant place in the next byte. In my case, I was implementing a FIFO buffer, so I needed this same functionality across groups of integers.

I opted on using an array as passing arbitrary variable names as the group is pretty much not-achievable. The functionality of BigSRR is similar to the Z80 instruction above but the shift ripples through the entire array resulting in the overall LSB or a zero moving into the overall MSB - probably best illustrated with the below graphic.



Shifting Left is the reverse action with the MSB or a zero moving into the right-most bit.

Only three elements are shown above, there is no practical limit on the number of elements in reality but the more elements, the longer any shift. The number of bits also affects execution time as confessed below.

As written below, the routines will allow a shift or rotate (depending on the value of the opt[ion] flag passed in). For now, they are supremely lazy, simply performing a 1 bit shift the number of times you request. i.e. if you shift the array 10 bits, it goes round the loop ten times. In that respect, I suppose they are faithful to the inspiration in that the Z80 didn't possess the barrel shifter of more modern CPUs that can shift any number of bits in a single cycle. I wanted to be clever and calculate a “cookie cut” on each array member but then it gets tricky when you shift more than 63 bits... so for now it is lazy but reliable.

Dependencies:

None

Examples:

BigSRL(1) Shift the entire array left 1 bit - same as BigSRL(1,0) BigSRR(10,1) Rotate the array right 10 bits. The ten bits that drop off the right hand end are shifted into the ten most significant bits

Some example code

```
    CPU 48

    Option Base 0 ' necessary
' variables to be rotated need to be in an array as we can't pass the names
of arbitrary variables.

    Dim Integer Vars=3,x' Vars is the number of 64 bit integers in the
array, change this as necessary
    Dim Integer B(Vars) ' B() is the buffer for our integers to shift. Byte
0, bit 63 is MSB, Byte[Vars], bit 0 is LSB
    'some demonstration code
    B(Vars)=1
    ?"In:"
    For x=0 To Vars
        ? Bin$(B(x),64)+" ";
    Next
    ?:?
    BigSRL(256,1)
    ?"Out:"
    For x=0 To Vars
        ? Bin$(B(x),64)+" ";
    Next
    ?
    BigSRR(253,1)
    ?"Out:"
    For x=0 To Vars
        ? Bin$(B(x),64)+" ";
    Next
    ?
```

The Subs

```
    'shift or rotate left B() bits. opt: 0=Shift, 1=rotate
    Sub BigSRL(bits as Integer,opt)
        Local Integer m,n,c
        For m=1 To bits
            c=(B(0) And &h8000000000000000)<>0
            For n=0 To Vars
                B(n)=B(n)<<1
                If n<Vars Then
```

```
                    If B(n+1) And &h8000000000000000 Then '
&B1000000000000000000000000000000000000000000000000000000000000000 63rd bit
                        B(n)=B(n) Or 1
                    EndIf
                EndIf
            Next
            If opt Then B(Vars)=B(Vars) Or c
        Next
    End Sub
    'shift or rotate right B() bits. opt: 0=Shift, 1=rotate
    Sub BigSRR(bits as Integer,opt)
        Local Integer m,n,c
        For m=1 To bits
            c=(B(Vars) And 1)*&h8000000000000000
            For n=Vars To 0 Step-1
                B(n)=B(n)>>1
                If n Then
                    If B(n-1) And 1 Then
                        B(n)=B(n) Or &h8000000000000000
                    EndIf
                EndIf
            Next
            If opt Then B(0)=B(0) Or c
        Next
    End Sub
```

```
\\
```