Program 1 (forward error correction block coder/transmitter)

Given a series of fifteen 11-bit message blocks in data mem[0:29], generate the corresponding 16-bit encoded versions and store these in data mem[30:59].

Input and output formats are as follows:

```
input MSW =    0   0   0   0   0 b11 b10 b09
      LSW =  b8  b7  b6  b5  b4  b3  b2  b1, where bx denotes a data bit

output MSW = b11 b10  b9  b8  b7  b6  b5  p8
      LSW =   b4  b3  b2  p4  b1  p2  p1 p16, where px denotes a parity bit
```

Example, to clarify "endianness": binary data value = 101_0101_0101

```
mem[1] = 00000101 -- 5 bits zero pad followed by b11:b9 = 00000_101
mem[0] = 01010101 -- lower 8 data bits b8:b1
```

You would generate and store:

```
mem[31] = 10101010 -- b11:b5, p8                = 1010101_0
mem[30] = 01011010 --  b4:b2, p4, b1, p2:p1, p16 = 010_1_1_01_0
```

```
p8 = ^(b11:b5) = 0;
p4 = ^(b11:b8,b4,b3,b2) = 1;
p2 = ^(b11,b10,b7,b6,b4,b3,b1) = 0;
p1 = ^(b11,b9,b7,b5,b4,b2,b1) = 1;
p16 = ^(b11:1,p8,p4,p2,p1) = 0;
```

```
var MSW0 = ((MSW & 0x7) << 5) | ((LSW >> 3) & 0x1e) | p8;
    LSW0 = ((LSW << 4) & 0xe0) |
```

STORE -> mem if needed

P1 on ARM?

lsl ~~asl~~

lsr

bic

and

orr

ldi