Pseudo-instructions are instructions in assembly language that are not part of the ARM instruction set. Instead, the assembler converts them into a small set of real ARM instructions. For each of the following instructions, convert them to an equivalent (VERY short) set of real ARM instructions. If you need a temporary variable, use X16 (a register reserved for the assembler as a temporary). You will also want to read about the MOVZ and MOVK instructions in Chapter 2.10 in the book.

<table>
<thead>
<tr>
<th>Pseudo-instruction</th>
<th>What it accomplishes</th>
</tr>
</thead>
<tbody>
<tr>
<td>LI X2, 32’big</td>
<td>X2 = big (note: solved below)</td>
</tr>
<tr>
<td>MOVE X1, X2</td>
<td>X1 = X2</td>
</tr>
<tr>
<td>BEQ X1, 16’small, L</td>
<td>if (X1 = small) go to L</td>
</tr>
<tr>
<td>ADDI X0, X2, 48’big</td>
<td>X0 = X2 + big</td>
</tr>
</tbody>
</table>

Note that constants have their bitwidths given as `bitwidth’const. If you need to access a portion of the constant, you can do that. For example, to implement “LI X2, 32’big”, you would write:

```assembly
MOVZ X2, big[31:16], LSL 16 // zero X2, set [31:16] properly
MOVK X2, big[15:0], LSL 0  // set bottom 16 bits

MOVE X1, X2
ADD X1, X2, X31           // X1 = X2 + 0. Many others work

BEQ

MOVZ X16, small           // X16 = small
CMP X1, X16               // Compare X1 to small
B.EQ L                    // if !=, go to L

ADDI

MOVZ X16, big[47:32], LSL 32
MOVK X16, big[31:16], LSL 16
MOVK X16, big[15:0], LSL  // X16 = 48’big
ADD X0, X2, X16
```
Translate the following C code to ARM assembly code. Use a minimum number of instructions. Assume that the value a, b, len, and D, are in registers X0, X1, X2, and X3 respectively, and D is an array of integers.

while (a<len) {
    D[a] = b + a;
    a += 1;
}

TOP:
    CMP    X0, X2   // Compare a with len
    BGE    DONE
    LSL    X4, X0, #3  // x4 = 8*a
    ADD    X5, X3, X4  // x5 = & D[a]
    ADD    X6, X1, X0  // x6 = b + a
    STUR   X6, [X5, #0]
    ADDI   X0, X0, #1  // a++
    B TOP