ARM PROGRAMMING

For the following C statement, write the corresponding LEGv8 assembly code. Assume that the variables `i`, and `j` are assigned to registers `X2` and `X3`, respectively. Assume that the base address of the arrays `A` and `B` are in registers `X6` and `X7`, respectively.

```
B[8] = A[i - j]
```

For the following C statement, write the corresponding LEGv8 assembly code. Assume that the variables `i`, and `j` are assigned to registers `X2`and `X3`, respectively. Assume that the base address of the arrays `A` and `B` are in registers `X6` and `X7`, respectively.

```
B[8] = A[i - j]
```

SUB X9, X2, X3 //X9 = i-j
LSL X9, X9, #3 //multiply the index by 8 to get bytes
ADD X10, X9, X6 //address of A[i-j]
LDUR X11, [X10, #0] //X11 = A[i-j]
STUR X11, [X7, #64] //B[8] = X11

Translate the following C code to LEGv8 assembly code. Use a minimum number of instructions. Assume that the values of `a`, `b`, `i`, and `j` are in registers `X0`, `X1`, `X10`, and `X11`, respectively. Also, assume that register `X2` holds the base address of the array D.

```
for(i=0; i<a; i++)
for(j=0; j<b; j++)
        D[4*j] = i + j;
```

Translate the following C code to LEGv8 assembly code. Use a minimum number of instructions. Assume that the values of `a`, `b`, `i`, and `j` are in registers `X0`, `X1`, `X10`, and `X11`, respectively. Also, assume that register `X2` holds the base address of the array D.

```
for(i=0; i<a; i++)
for(j=0; j<b; j++)
        D[4*j] = i + j;
```

```
          ADD X10, XZR, XZR //not sure if this is necessary but….
          ADD X11, XZR, XZR
iloop:    SUB X12, X0, X11
          CBZ X12, done
jloop:    SUB X13, X1, X11
          CBZ  X13, exitj
          ADD X14, X10, X11 //i + j;
          LSL   X15, X11, #2  //4j
          LSL   X15, X15, #3 //byte offset
          ADD X16, X15, X2  //address of D[4j]
          STUR X14, [X16, #0]
          ADD X11, X11, #1
          B jloop
exitj:    ADD X10, X10, #1
          ADD X11, XZR, XZR
          B iloop
done:
```

Translate function `f` into LEGv8 assembly language. If you need to use registers `X10` through `X27`, use the lower-numbered registers first. Assume the function declaration for g is `"int g(int a, int b)"`. The code for function `f` is as follows:

```
int f(int a, int b, int c, int d){
return g(g(a,b), c+d);
}
```

```
f:      //x0=a, x1=b, x2=c, x3=d
        //prepare a stack frame for a, b, c, d and return address
        SUB X28, X28, #48   //has to be a multiple of 16
        STUR X0, [X28, #32]
        STUR X1, [X28, #24]
        STUR X2, [X28, #16]
        STUR X3, [X28, #8]
        STUR X30, [X28, #0]
        //call g(a,b), don't need to change X0 and X1
        BL g
        //X2 has the result from g(a,b), save it
        ADD X10, X2, XZR
        //restore values from stack
        LDUR X2, [X28, #16]
        LDUR X3, [X28, #8]
        ADD X11, X2, X3 //c+d
        //stack has  f's information on top, so don't need to prepare it again
        //prepare for second call to g
        ADD X0, X10, XZR //put the result from g(a,b) into X0
        ADD X1, X11, XZR  //put c+d into X1
        BL g
        //X2 has the result from g(g(a,b), c+d), it needs to be returned in X4
        ADD X4, X2, XZR
        //restore values and stack
        LDUR X0, [X28, #32]
        LDUR X1, [X28, #24]
        LDUR X2, [X28, #16]
        LDUR X3, [X28, #8]
        LDUR X30, [X28, #0]
        ADD X28, X28, #48
        BR X30
```