1. Compute: \( zz = aa*(bb+cc) – (dd*35) \);
   Place \( aa, bb, cc, dd \) in your code area, so that you can provide initial values with “DCD” assembler directives. Place \( zz \) in the data area, so that you can write the result to it. The final debug window should show the final value of \( zz \) in memory.

2. Implement the assembly language equivalent of the following C code, to exercise program control statements.

   ```c
   if ((mm – nn) < 15) {
       kk = jj – 500;
       xx = 7;
   } else {
       kk = cc + 180;
       xx = -7;
   }
   ```

   Place \( mm, nn, jj, cc \) in the code area, with initial values defined by DCD directives, and place \( kk \) and \( xx \) in the data area. Circle the values of \( kk \) and \( xx \) in the final debug window. Execute this program twice – once to show execution for the tested condition true, and once to show execution for the tested condition false.

3. Implement the assembly language equivalent of the following C code, to exercise memory addressing modes to handle arrays. Place arrays \( aa \) and \( bb \) in the code area, with initial values. These arrays each have 15 elements. Place variable \( i \) and array \( zz \) in the data area. Circle the final values of \( i \) and \( zz \) in the debug window.

   ```c
   for (i = 0; i < 15; i++)
       zz[i] = aa[i] – bb[i] + 5;
   ```