*Note*: Assembly subroutines must comply with the ARM Architecture Procedure Call Standard (AAPCS) standard (about parameter passing, returned value, callee-saved registers).

In all subroutines, you can assume that there is no overflow with 32-bit operations.

**Specification 1** (4 points). The truncation of the Taylor series for the cosine function to a second-order approximation is:

In equation (1), both the angle *y* and the value cos *y* are real numbers. They can be expressed as integer numbers to the detriment of precision. In particular, both angle *y* and value cos *y* are multiplied by 128 (i.e., each term in (1) is multiplied by 128 and the substitution *x* = 128 *y* is applied). We obtain:

Write the cosine subroutine that computes the value of 128 🞝 cos *x* according to equation (2).

The subroutine receives in input the integer values *x*.

You can not use the division operation; instead, use a right shift to divide a number by power of 2.

The division (with right shift) must be computed after computing the numerator *x*2 in (2).

**Specification 2** (8 points). The truncation of the Taylor series for the sine function to a first-order approximation is sin *y* = *y*, i.e.

by substituting the real value *y* with the integer number *x* as explained in specification 1. However, this approximation is accurate only for small angles. In particular, we use the approximation if -0.5 rad < *y* < 0.5 rad, i.e., -64 < *x* < 64.

For bigger angles, we can use the double-angle formula:

With the usual substitution, equation (4) becomes:

Write the doubleAngleSine subroutine that computes the value of 128 🞝 sin *x* as follows:

* according to equation (3) if -64 < *x* < 64
* according to equation (5) otherwise. In this case, doubleAngleSine calls the cosine subroutine passing *x*/2, and then it calls recursively doubleAngleSine passing *x*/2. The results returned by both functions are multiplied with each other and then multiplied by 2/128 (note: use a right shift instead of the division).  
  Note: the value returned by the cosine subroutine must be saved in a call-preserved register (according to AAPCS standard), in order to not be overwritten during the execution of the next doubleAngleSine call.

**Specification 3** (6 points). The truncation of the Taylor series for the cosine function to a second-order approximation, as expressed in equation (1) and (2), is accurate only for small angles. In particular, we use the approximation if -1 rad < *y* < 1 rad, i.e., -128 < *x* < 128.

For bigger angles, we can use the double-angle formula:

With the usual substitution, equation (6) becomes:

Write the doubleAngleCosine subroutine that computes the value of 128 🞝 cos *x* as follows:

* according to equation (2) if -128 < *x* < 128. In this case, doubleAngleCosine calls the cosine subroutine passing *x*.
* according to equation (7) otherwise. In this case, doubleAngleCosine calls the doubleAngleSine subroutine passing *x*/2. Then, the result is squared, multiplied by 2/128 (note: use a right shift instead of the division) and subtracted from 128 as indicated in equation (7).

**Specification 4** (2 points).

Write the recursiveCosine subroutine. The body of the subroutine is copied and pasted from the doubleAngleCosine subroutine. Then, apply the following changes:

* labels are renamed, to avoid duplication of labels
* call to doubleAngleSine subroutine is replaced with the call to recursiveSine subroutine.

Similarly, write the recursiveSine subroutine. The body of the subroutine is copied and pasted from the doubleAngleSine subroutine. Then, apply the following changes:

* labels are renamed, to avoid duplication of labels
* call to cosine subroutine is replaced with the call to recursiveCosine subroutine
* call to doubleAngleSine subroutine is replaced with the call to recursiveSine subroutine.