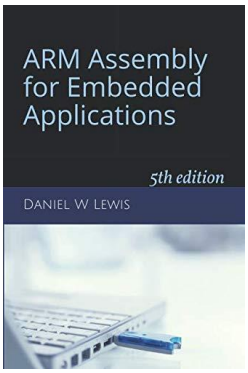


Programming Lab 6A

Spinning Cube

Topics: Two-dimensional subscripting, nested loops, calling C functions from assembly

Click to download Lab6A-Main.c

Prerequisite Reading: Chapters 1-6

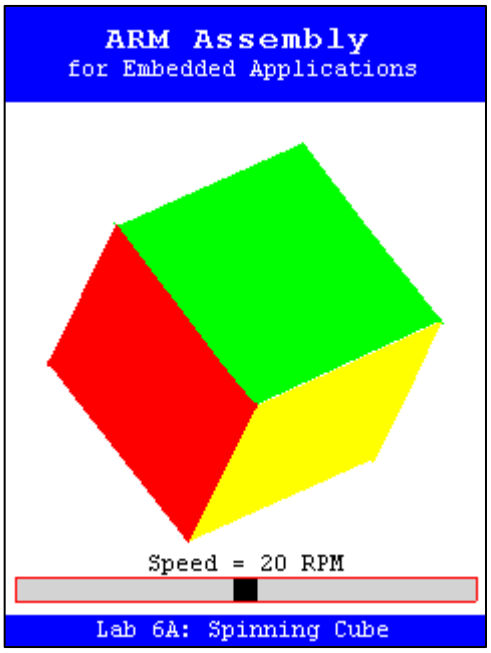
Revised: June 22, 2021

***Background*[1]:** In 3D computer graphics, object surfaces are modeled as a collection of triangles. (E.g., each face of a cube may be modeled using two triangles.) Each vertex of a triangle is represented as a vector $V = [V_x, V_y, V_z]$, where V_x, V_y and V_z are the usual Cartesian coordinates in 3-space. Linear algebra and matrix multiplication are used to modify the position of vertices and thus the position and orientation of objects. For example, multiplying matrix M^x (given below) times vector V creates a new vector V' that corresponds to rotating the position of the vertex represented by vector V around the x-axis by θ radians:

$$V' = \begin{bmatrix} V'_x \\ V'_y \\ V'_z \end{bmatrix} = M^x \times V = \begin{bmatrix} 1.0 & 0.0 & 0.0 \\ 0.0 & \cos\theta & -\sin\theta \\ 0.0 & \sin\theta & \cos\theta \end{bmatrix} \times \begin{bmatrix} V_x \\ V_y \\ V_z \end{bmatrix}$$

The product of two 3x3 matrices is another 3x3 matrix. Given two matrices M^x and M^y that rotate vertices around the x and y-axis respectively, the product $M^{xy} = M^x M^y$ is a single matrix that combines both rotations, where the value in row r, column c of M^{xy} is given by:

$$M^{xy}_{r,c} = \sum_{k=0}^{k=2} M^x_{r,k} \times M^y_{k,c}$$



Assignment: The main program will compile and run without writing any assembly. However, your task is to create an equivalent replacement in assembly language for function `MatrixMultiply` found in the C main program. The original C version has been defined as "weak" so that the linker will automatically replace it in the executable image by the one you create in assembly; you do not need to remove the C version.

```
void MatrixMultiply(int32_t A[3][3], int32_t B[3][3], int32_t C[3][3]) ;
```

Function `MatrixMultiply` implements matrix multiplication based on the following pseudo-code:

```
for row ← 0 to 2 do:
   for col ← 0 to 2 do:
      set A_row,col ← 0
      for k ← 0 to 2 do:
         A_row,col ← MultAndAdd(A_row,col, B_row,k, C_k,col)
```

For example: $A_{0,1}$ is the sum of products of corresponding elements from row 0 of matrix B and column 1 of matrix C:

$$\begin{bmatrix} A_{0,0} & \boldsymbol{A_{0,1}} & A_{0,2} \\ A_{1,0} & A_{1,1} & A_{1,2} \\ A_{2,0} & A_{2,1} & A_{2,2} \end{bmatrix} = \begin{bmatrix} \boldsymbol{B_{0,0}} & \boldsymbol{B_{0,1}} & \boldsymbol{B_{0,2}} \\ B_{1,0} & B_{1,1} & B_{1,2} \\ B_{2,0} & B_{2,1} & B_{2,2} \end{bmatrix} \times \begin{bmatrix} C_{0,0} & \boldsymbol{C_{0,1}} & C_{0,2} \\ C_{1,0} & \boldsymbol{C_{1,1}} & C_{1,2} \\ C_{2,0} & \boldsymbol{C_{2,1}} & C_{2,2} \end{bmatrix}$$

Test your implementation of the `MatrixMultiply` function using the C main program. Note that function `MatrixMultiply` should call function `MultAndAdd` that is implemented in the C source code file[2]. If your code is correct, the display should display a rapidly spinning cube like the image above. Use the blue pushbutton to pause or the slider to change the speed. Any errors detected in your function will be displayed as white text on a red background.

---

[1] https://en.wikipedia.org/wiki/Transformation_matrix

[2] **IMPORTANT:** Don't replace function MultAndAdd with integer multiply and add instructions; we've hidden the fact that it actually uses floating-point to do arithmetic. Just code your solution assuming that the arrays hold 32-bit integers.