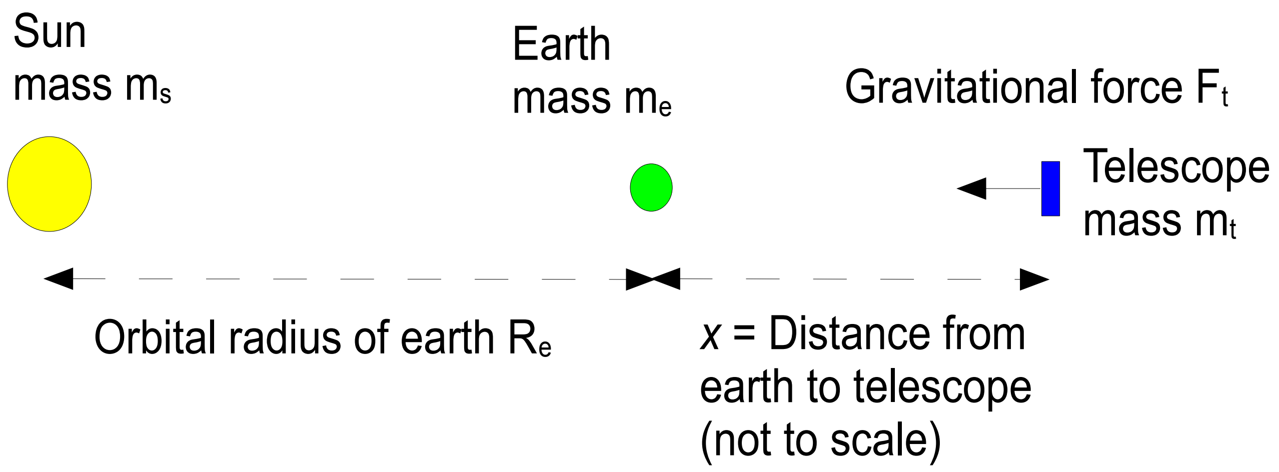
CS2240 Spring 2023
Programming Project #3 – The James Webb Space Telescope
Due: Friday, April 21, 2022, 11:59pm

The launch of the James Webb Space Telescope in 2021 has provided direct observations of the history of the universe unprecedented in human history. For this telescope to be able to make these observations, it must be physically located at the "L2 Lagrange Point", a unique point in space with respect to the sun and the earth, and the last part of its launch process was a rocket-propelled journey out to this point. At the L2 point, the gravitational forces on it from the earth and sun add up to just the right force so that in its orbit around the sun, the telescope stays at a fixed point in the sky as seen from earth. Your coding task for this project is to determine how far away from earth is this singular L2 point, which is the unknown distance *x* in this diagram:


Sun
mass m_s
Earth
mass m_e
Gravitational force F_t
Telescope
mass m_t
Orbital radius of earth R_e
x = Distance from earth to telescope (not to scale)

The mathematical problem needed to find the distance to this exact combination point of gravitational forces involves finding the root of a 5th order polynomial, and cannot be solved analytically. Numerical computational methods with floating point arithmetic must be used to find *x* and locate the telescope properly. By applying Newton's laws of gravitation to this geometry, the value of *x* at the L2 point will be the *root* of the following function, that is, the value of *x* that makes the function

$$f(x) = q^3 p^2 m_s - p^2 m_s - q^2 m_e$$

go to zero, where

$$p = \frac{x}{R_e} \qquad q = 1+p = 1+\frac{x}{R_e}$$

m_s = The mass of the sun, or 2.0e30 kilograms
m_e = The mass of the earth, or 5.98e24 kilograms
R_e = The radius of earth's orbit around the sun = 1.50e11 meters

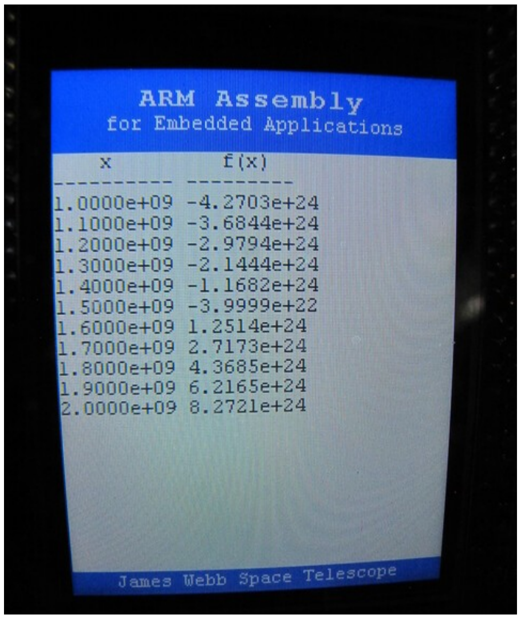
Since we will use the orbital radius in units of meters, finding the value of *x* that satisfies this equation will give us the distance *x* in units of meters as well.

Download the C code file Project3-JWSTMain.c and the template assembly source file Project3-JWST.s. Your project code will resemble the lab exercises with two source code modules, a main program in C and an assembly module. For this project the assembly module will provide one function named `JWST` called by the C code, and this function will perform all the arithmetic computation of the values of the function *f(x)* as defined above versus 11 stepped values of the parameter *x*. Your function is to be called with four arguments: a pointer to a `float` array for the *x* values, a pointer to a `float` array for the *f(x)* values, the first value of *x* and the last value of *x*. Your function must fill in 11 values of the parameter *x* and the function *f(x)* into the two arrays pointed to by the first and second arguments, with each function valued computed from a parameter *x* that must be stepped from a first value up to **and including** the last value, for 11 linearly stepped values. The main C program will then display a list of the computed values of *f(x)* versus *x* on the Discovery Board screen. Note in the main C program the two constant definition lines near the top that set the initial first and last values for *x* that your function will be called with to 1.0e9 meters and 2.0e9 meters.

Use the following pseudocode as a template for your function:

```
Load the value of the constants $M_s$, $M_e$, and $R_e$ from memory into
registers
Compute $x_{inc} = \frac{(x_{last} - x_{first})}{10.0}$
Set $x = x_{first}$
Set $i = 0$
While $i$ < 11 do
  Compute $p$
  Compute $q$
  Compute $f(x)$
  Store $x$ into the x array, index $i$
  Store f(x) into the f array, index $i$
  Increment $i$
  Set $x = x + x_{inc}$
End while
Return
```

Make use of your code from Lab #10 as example floating point arithmetic for the calculations. Make use of your code from Lab #9 and for Lab #10 addressing elements in an array whose pointer is specified from the C main program as a function argument. The arrays in this C program are of float values, with each element requiring four bytes like the array in Lab #10, so write your memory addressing accordingly. Hint: Remember the helpful '`SVC #0`' and '`SVC #1`' debugging instructions available with your Discover Board library! That can be inserted anywhere in your assembly code any time you need to clarify exactly what is in the core and floating point registers.



```
      ARM Assembly
for Embedded Applications
    x          f(x)
---------- ----------
1.0000e+09 -4.2703e+24
1.1000e+09 -3.6844e+24
1.2000e+09 -2.9794e+24
1.3000e+09 -2.1444e+24
1.4000e+09 -1.1682e+24
1.5000e+09 -3.9999e+22
1.6000e+09 1.2514e+24
1.7000e+09 2.7173e+24
1.8000e+09 4.3685e+24
1.9000e+09 6.2165e+24
2.0000e+09 8.2721e+24
 James Webb Space Telescope
```

Build and run your project code. Make sure that initially it generates the listing on the Discovery Board screen matching the display shown here for the initial values of 1.0e9 and 2.0e9 for XFIRST and XLAST.

Note that the function *f(x)* changes its sign from negative to positive between the *x* values of 1.5e9 and 1.6e9. For continuous functions such as this polynomial, that means that the root of *f(x)*, which is the *x* value at the L2 Lagrange point that makes *f(x)*=0, must lie somewhere between 1.5e9 and 1.6e9 meters. Edit the main C program and change the constants setting XFIRST and XLAST to these new bracketing values. Rebuild and rerun your code. This will generate a new table of *f(x)* values. Pick the two new values of *x* that again bracket the root of the function, change the constants in the C code, and rerun. In only a few steps like this you should find a value of *x* sufficiently close to the root of the function. For the purposes of locating the L2 point, consider an *x* value that reduces the absolute value of the function to be less than 1.0e19 to be close enough to the point in practice. Upload your code to Canvas, a screen photo of your final table of function values, and the value of *x* you are claiming as your solution. Make sure, however, that any debugging breakpoints you inserted for debugging your code have been removed in the version you submit.