Assignment: Factoring  
Due December 7, 2005

Assignment

Write a function that can be used with a C++ program which will find the smallest prime factor of a given 32-bit unsigned integer parameter. Your function should have the signature: `unsigned long Factor(unsigned long)`.

For instance, `Factor(35)` should return 5, and `Factor(4000000001)` should be 47.

Since finding prime factors can be quite time consuming, this is a good example of a fairly practical application of assembly language. Along with the speed increase that comes from coding directly in assembly language, you should take care to use as efficient an algorithm as you can think of. You might, for instance, notice that except for 2, prime factors are all odd. Also, unless a number is prime, its smallest factor is no larger than its square root. You may be able to come up with other optimizations—this subject has been extensively studied, and many quite effective algorithms developed. Finally, you may wish to consider one of the divide instructions that requires a 68020 or better, like `divul.l`, since all modern Motorola-based machines will recognize these instructions.

Procedure

Use the Code Warrior development environment again for this project.

First, download the C++ driver program `factor.cpp` to your account from:

This driver program includes a test to make sure your function returns the correct answer, and a timing section to measure how quickly your function executes.

On the Mac, begin by setting up a new project, as you did in the last assignment. When the project has been created, use Project/Add Files to make `factor.cpp` part of your project, under Sources.

Testing

The C++ program you downloaded has a driver and a simple C++ version of `Factor()` to test your function. In addition to using your function to factor numbers, it compares your results to the `ZFactor()` results, and does a timing to show you how fast (or slow...) your routine is. Try to achieve speeds as much faster than `ZFactor’s` as you can.

Test of course with a variety of test values. Here is one sample run from my old G3 powerbook, using a highly optimized version of Factor written, of course, in assembly language and using a fairly sophisticated factoring algorithm. Hopefully, newer Macintoshes will produce faster runtimes, but note that it is really the ratio of the C++ to assembly speeds we are most interested in.

Enter a test value (0 to quit): 1234567  
1234567 = 127 * 9721

Timing using 150000 repetitions  
1234567 was factored by Factor() in about 0.0263333 milliseconds  
C++ reference using ZFactor(): 0.122444 milliseconds
Enter a test value (0 to quit): 123456789
123456789 = 3 * 3 * 3607 * 3803

Timing using 8571 repetitions
123456789 was factored by Factor() in about 0.328628 milliseconds
C++ reference using ZFactor(): 2.0301 milliseconds

Enter a test value (0 to quit): 1234567899
1234567899 = 3 * 3 * 3 * 3 * 109 * 139831

Timing using 60000 repetitions
1234567899 was factored by Factor() in about 0.0544444 milliseconds
C++ reference using ZFactor(): 0.272222 milliseconds

Enter a test value (0 to quit): 4123456789
4123456789 = 191 * 491 * 43969

Timing using 33333 repetitions
4123456789 was factored by Factor() in about 0.0865009 milliseconds
C++ reference using ZFactor(): 0.490505 milliseconds

Enter a test value (0 to quit): 0

**Turning it in**

On or before the due date, turn in to me a listing of your `Factor()` assembly language function (not the rest of the driver, etc.) and a sample run showing several tests, along with a disk or email enclosure (rayz@cs.csustan.edu) containing the complete source code so I can test it. On the disk, write your name and the name of your source file so I can find it easily.