EE467/567: Advanced Microprocessors

Project #1

Due: *Wednesday 22 February 2012 by 11:59 PM*

Submission Requirements*: Do not zip the files together, rename the files <lastname>p1q1.cpp, <lastname>p1q2.cpp and <lastname>p1q3.cpp and send it* ***separately*** *to kodi@ohio.edu*

Use the C++ templates for the Inline Assemble programs for problems 1 (*p1q1.cpp*), 2 (*p1q2.cpp*), and 3 (*p1q3.cpp*) as given on my website. Comment your assembly code and make sure your syntax is correct!! Code that does not run results in the deduction a significant amount of points. For each of the problems below write a paragraph that describes your approach to the problem (this description can be included in the top of the C++ program).

1. Suppose we have a signal, "x[n]" which contains a noisy sine wave. A digital filter could be used to reduce the noise content of "x[n]". One such filter is the 8-point averager. This filter is referred to as a low-pass filter and removes part of the noise. The mathematical equation for an 8-point averager is:

$$y[n] = \frac{1}{8}\left(x[n] + x[n-1] + x[n-2] + x[n-3] + x[n-4] + x[n-5] + x[n-6] + x[n-7]\right)$$

   Write an inline assembly program that implements this filter using ***fixed-point*** arithmetic. The array "x" contains 100 8-bit signed numbers. The values are given in the "CPP" program provided on my website (*p1q1.cpp*). Only execute this filter for n = 7, 8 … 99. In other words find: $y[7]$, $y[8]$, $y[9]$, …,$y[99]$. Set the values $y[0]$ through $y[6]$ to 0.

2. Suppose we have a signal, "x[n]" which contains a noisy sine wave. A digital filter could be used to reduce the noise content of "x[n]". One such filter is the 8-point averager. This filter is referred to as a low-pass filter and removes part of the noise. The mathematical equation for an 8-point averager is:

$$y[n] = \frac{1}{8}\left(x[n] + x[n-1] + x[n-2] + x[n-3] + x[n-4] + x[n-5] + x[n-6] + x[n-7]\right)$$

   Write an inline assembly program that implements this filter using ***floating-point*** arithmetic. The array "x" contains 100 floating-point numbers. The values are given in the "C" program provided on my website (*p1q2.cpp*). Only execute this filter for n = 7, 8 … 99. In other words find: $y[7]$, $y[8]$, $y[9]$, …,$y[99]$. Set the values $y[0]$ through $y[6]$ to 0.0.

3. Given an array "x" of 100 16-bit signed numbers stored in memory, write an inline assembly program to generate two new arrays from the given array such that one array 'P' consists of all positive numbers and the other 'N' contains all the negative numbers. Return the array of positive numbers and the array of negative numbers in two different arrays. Furthermore, the subroutine should "return" the number of zero-values it found. The array "x" is given in the "CPP" program file (*p1q3.cpp*).