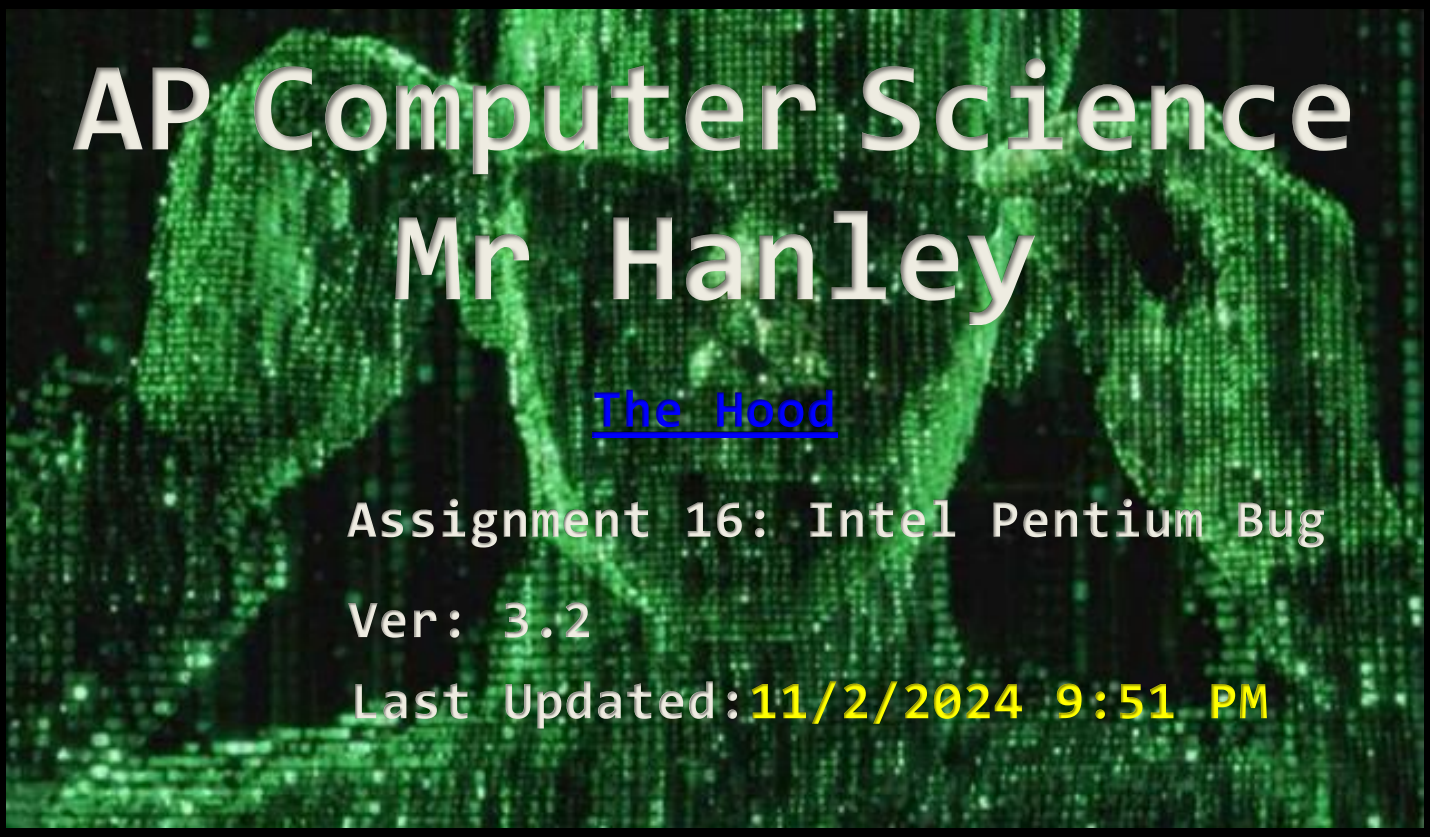


Assignment 16:Intel Pentium Bug

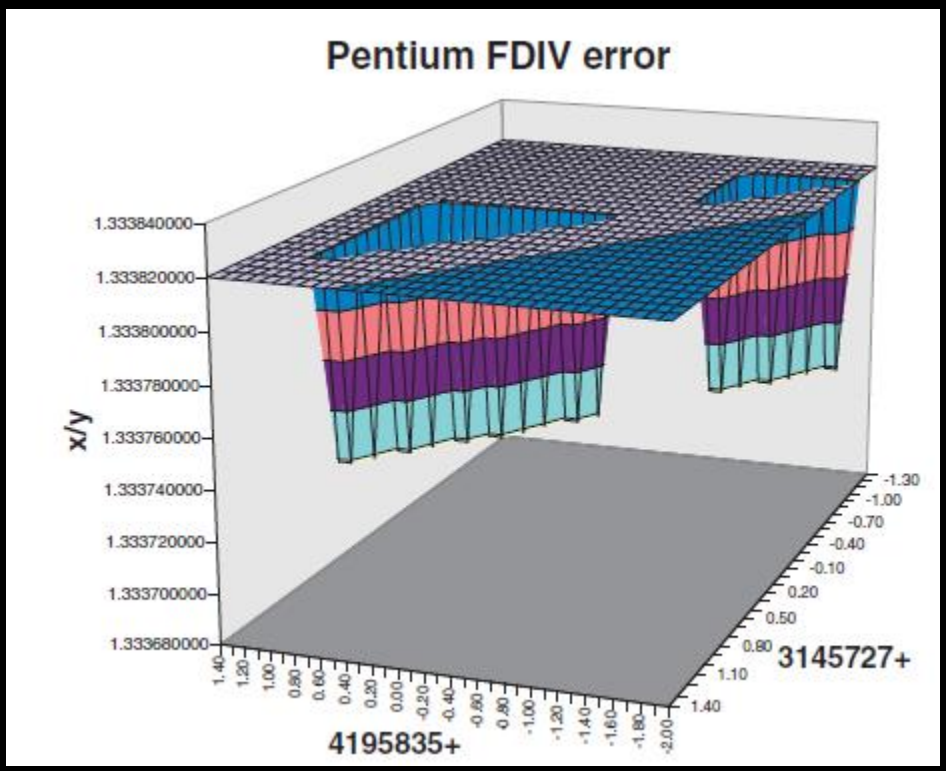
The Pentium Floating-Point Bug

In 1994, Intel Corporation released what was then its most powerful processor, the first of the Pentium series. Unlike previous generations of Intel's processors, the Pentium had a built-in floating-point unit. Intel's goal was to compete aggressively with the makers of higher-end processors for engineering workstations. The Pentium was an immediate success.

In the summer of 1994, Dr. Thomas Nicely of Lynchburg College in Virginia ran an

extensive set of computations to analyze the sums of reciprocals of certain sequences of prime numbers. The results were not always what his theory predicted, even after he took into account the inevitable roundoff errors. Then Dr. Nicely noted that the same program did produce the correct results when run on the slower 486 processor, which preceded the Pentium in Intel's lineup. This should not have happened. The roundoff behavior of floatingpoint calculations had been standardized by the Institute of Electrical and Electronics Engineers (IEEE), and Intel claimed to adhere to the IEEE standard in both the 486 and the Pentium processors.

Upon further checking, Dr. Nicely discovered that indeed there was a very small set of numbers for which the product of two numbers was computed differently on the two processors. For example, $4{,}195{,}835 = ((4{,}195{,}835 / 3{,}145{,}727) \cdot 3{,}145{,}727)$ is mathematically equal to 0, and it did compute as 0 on a 486 processor. On a Pentium processor, however, the result was 256.
As it turned out, Intel had independently discovered the bug in its testing and had started to produce chips that fixed it. (Subsequent versions of the Pentium, such as the Pentium III and IV, are free of the problem.) The bug was caused by an error in a table that was used to



This graph shows a set of numbers for which the original Pentium processor obtained the wrong quotient.

speed up the floating-point multiplication algorithm of the processor. Intel determined that the problem was exceedingly rare. They claimed that under normal use a typical consumer would only notice the problem once every 27,000 years. Unfortunately for Intel, Dr. Nicely had not been a normal user. Now Intel had a real problem on its hands.

It figured that replacing all the Pentium processors that it had already sold would cost it a great deal of money. Intel already had more orders for the chip than it could produce, and it would be particularly galling to have to give out the scarce chips as free replacements instead of selling them. Initially, Intel's management offered to replace the processors only for those customers who could prove that their work
required absolute precision in mathematical calculations. Naturally, that did not go over well with the hundreds of thousands of customers who had paid retail prices of $700 and more for a Pentium chip and did not want to live with the nagging feeling that perhaps, one day, their income tax program would produce a faulty return. In the end, Intel gave in to public demand and replaced the defective chips, at a cost of about 475 million dollars.

You are going to take the role of a US Senator. Intel has a problem with their processors and you have to decide what to do in the best interest for society. Should you pressure Intel to replace all of the defective processors? Should you allow them to have users who can prove they will encounter the error before replacing? Or perhaps allow Intel to offer the option of a cash rebate as mentioned in Big Java. Take a position and argue for what you would do.

Then respond to what you would do if you had a defective Pentium and were offered the option of cash or replacement. Use at least 380 words.
There should be correct grammar, no spelling errors and no run on sentences. Before you submit your typed essay, be sure to read it out loud to yourself or someone else.

RUBRIC	
Well argued and thought out	30
-2 for each spelling error	
-2 for each grammatical/run on sentence	
TOTAL	30


Artificial_Intelligence
Dynamic_Memory
Waterfall_Model
Nested_Loops
Stacks
Short_Circuit_Evaluation
Insertion_Sort
Binary_Search
Inheritance
Files
Linear_nlogn
Extreme_Programming
Client_Server
Hong_Kong_Fooey
Video_Games
String
Recursion
Search
File_Processing
Arrays
Selection_Sort
Big_O_Notation
2D_Arrays
Hangman
ArrayList