==================================================

Student Name：彭敬樺 Assignment：1

Student ID：0416106

Student Email: wilbert.phen@gmail.com

**Claim: I worked on my own. ( YES / NO )**

**Introduction**

Word Count: 185 words

Assignment 1 consists of assembly program to compute the estimation of pi. The pi estimation in this assignment will required us to first understand its mathematical representation using Taylor Series. Taylor Series is a representation of a function as an infinite sum of terms that are calculated from the values of the function’s derivatives at a single point. This series formed by taking some initial terms of the Taylor series. However, the series used must be convergence to a finite number so that the result is defined. Using this methodology, we will have a very good approximation of the irrational number pi. The problem arises when we try to pinpoint the exact amount of pi, the reason is because we can only compute finite amount of numbers in a computer program, not infinite numbers. Computer memory is limited to a finite amount, so calculating infinite number is an impossible task for a computer. Therefore, for this assignment the number of terms will be limited to initial 100,000 first terms. By using huge amount of terms, the errors arises from this estimation method could be minimizes.

**Program description**

Word Count: 208 words

This program is titled “Pi Calculation”, created by Wilbert (彭敬樺) at 27th February 2017. To assist the program, some file is included: Irvine32.inc and macros.inc. The program starts with printing out the basic information of its creator.

After it is printed, the user will be prompted to input an integer ranging from 1 to 100,000 which represents the number of terms the user desired to compute the estimation of pi. The larger the integer given by the user, the more accurate the estimation given by this program will be towards pi. If the user try to input an integer which is below 0, then the program will request re-input of the integer. If the integer is larger than 100,000 and still below overflow value, then the number will be interpreted as 100,000 which will be the limit of terms accepted in this program.

The result of the approximation will be printed directly to the console interface and will continue to request next input from the user to calculate the next request. This phenomenon will repeats continuously if the user desired it to be. To exit the program, user could input the number 0(zero) which will be translated as the command to finish the code.

**Methodology**

Word Count: 365 words

The program contains in main.asm consists of code and data parts. In the data parts, some important variable is declared and initialized. This is done to reduce the chance of any misunderstanding in the value of those variable before starting the program. Variable “ZERO”, “ONE”, TWO” and “FOUR” is declared so that the floating point operation could be done without using an immediate number. The important variable of the programs lies on variable “MAX”, “ANS”, “NUM, and “SIGN”. As the name described, the consequent variable have their own responsibility to maintain a value for that purpose.

In this program, the function “mWrite” is used to directly print the desired string onto console interface which is much easier and practical to use than “WriteString”.

All parts of the program except the parts, which prints out basic information is contained inside a logical loop that runs forever. User’s input will act as command for what the program should do. If 0(zero) in inputted, then the program will jump to the address that contains the instruction to quit and terminate the program.

After appropriate input has been accepted, the program will continue to re-initialize every used variable to diminish its past value that it could retain from previous operation. After this step, next parts is the calculation parts.

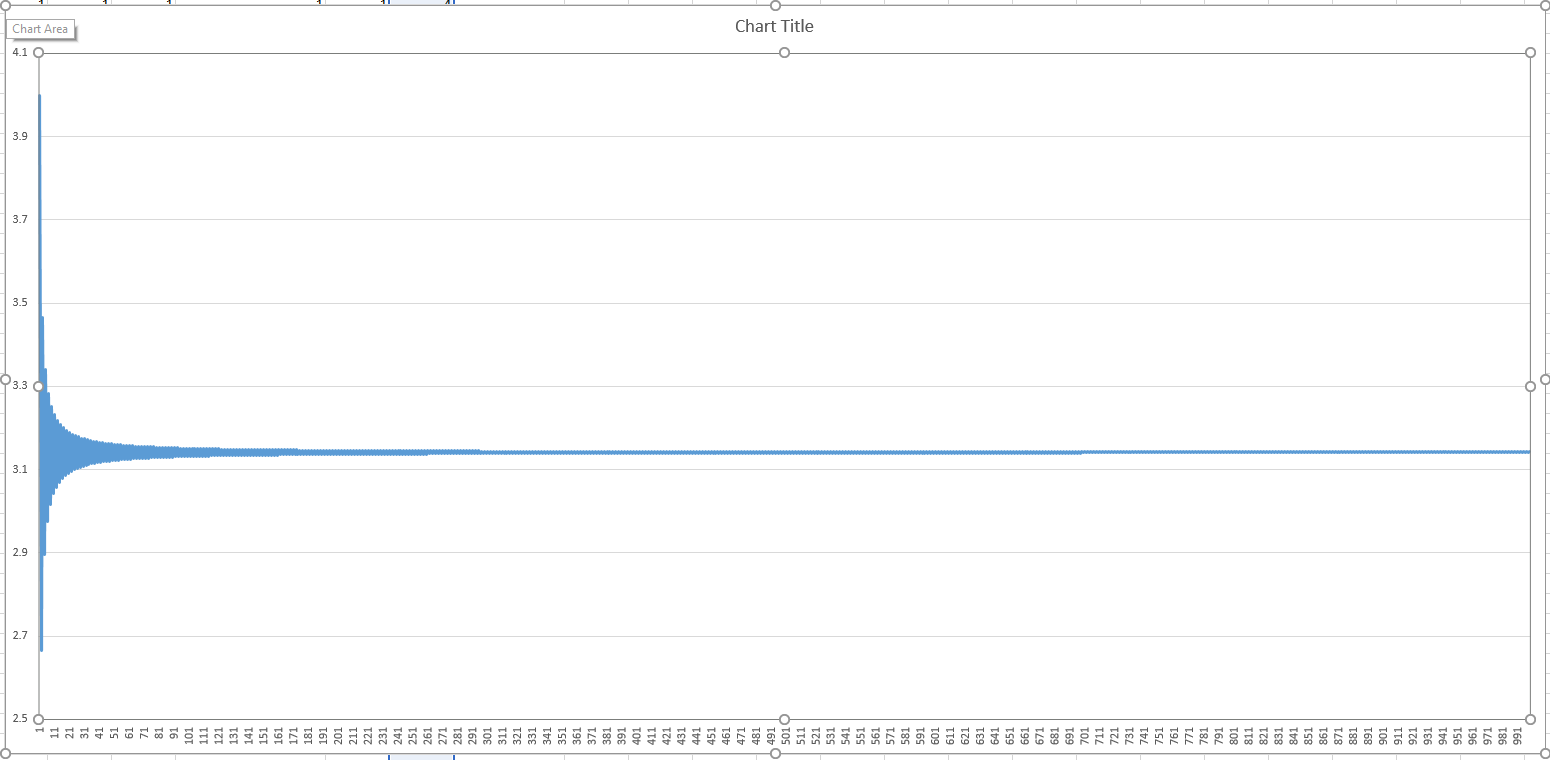
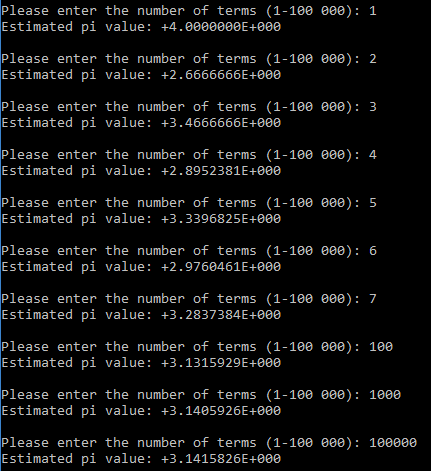
First, we calculate the value of this terms we are on by dividing one by it divisor. Next, the sign of this term will be changed from its previous attribute, which will result in the sign changing for every neighboring term. The current divisor has to be changed too. It is added with 2 to change it into the value of divisor that next term will need. This process is repeated for n times which is inputted by user.

The calculation process above will have its result stored in the variable “ANS”. However, this “ANS” has not have its correct value yet. The value we have calculated is to approximate the value of pi divided by 4(four). To get the final answer, the variable “ANS” and “FOUR” will be loaded into the floating point register to be multiplied. This final answer will be the number printed on user’s console interface.

**Experiments**

Word Count: 238 words

The graph below shows the estimated value of pi using different total of terms. In the graph, it will only be included until the 1000th term. Since the corresonding next line will be too small to be perceived. We can see that actually pi could reach its high precision in only less than 1000 terms of computation. As we know, the more terms included in the calculation, the more precise the number will be.



The second pictures show the estimated pi value correspond to its number of terms inputted by its user. The first terms is the max number of pi, therefore this value is the upper bound for this infinite function polynomial. If the terms included in process is 2, then the number that we also seen in the picture is the lower bound of the function polynomial.

The number of pi is actually **3.14159265359….**, which has been sliced to show only precision until a few digits behind comma.

Just as seen in the picture, the number of terms needed to be precise enough to reach pi is not much. After 1,000 terms, the first 2 digits after comma has reached its absolute precision which will not change anymore even if more terms is added. After 100,000 terms, the first 4 digits has reach the actual value of pi. The actual pi in this report is gained from calculator in “Google” search engine that provide high precision.

**Conclusion**

Word Count: 185 words

In conclusion, an irrational number could be approximated using an infinite series which approach the actual number however never the same. This method commonly used in computers that could calculate faster than average human. Many irrational number has its own infinite polynomial that could represents them. Therefore, by completing this pi calculation, many irrational number can now be easily calculated in the future.

In this program, I also learnt many good experience from debugging my own program. This first assignment provides good amount of practices to complete it. Some mistakes made:  
 - Treating unsigned and signed number as the same  
 - Misused of function such as jump less than as equal to “<=”  
 - Forget to initialize the variable after each of them is used in a loop  
 - Trying to load a floating-point register using an immediate value.

Implementing an assembly based program is more challenging than other high level programming language which have more connection into normal human conversation. However, by this course and this assignment, my understanding towards how a function or protocol is implemented in higher level language has become better.