

Student Name:_____ 楊林彬_____ Assignment: __HK1__

Student ID: _____0540137_____

Student Email: _____2830406835@qq.com_____

Claim: I worked on my own. (YES / NO) Must tick one.

If this is not your own work, your score is zero.

DO NOT CHEAT in programming. You should learn on your own.

1. Introduction (at least 150 words) [15%]

Word Count: _____ (must filled, or zero point)

This is my first homework for Assembly Language during this semester. The requirement is that students use X86 assembly language to write a program which applies the Taylor series method to calculating PI. To finish this assignment, we should know how to use FPU set and some basic commands in assembly language like jnz, loop, etc. In terms of the specific way to build the program, below I give my detailed explanation. In the first part, I give a general introduction on the main structure of my program and then the design ideas are present in the Methodology part. As for the third part, I show some screen shots, which record the running states of my program, there are also some tables recording the computed values with different input terms. I use Python to finish this assignment additionally, and use python as an important tool to analyse my ASM program. Apart from these, I show how I debug my program and make it better in this part. In the Conclusion part, I briefly summarize what I have learned from this assignment.

2. Program description (at least 200 words) [20%]

Word Count: _____ (must filled, or zero point)

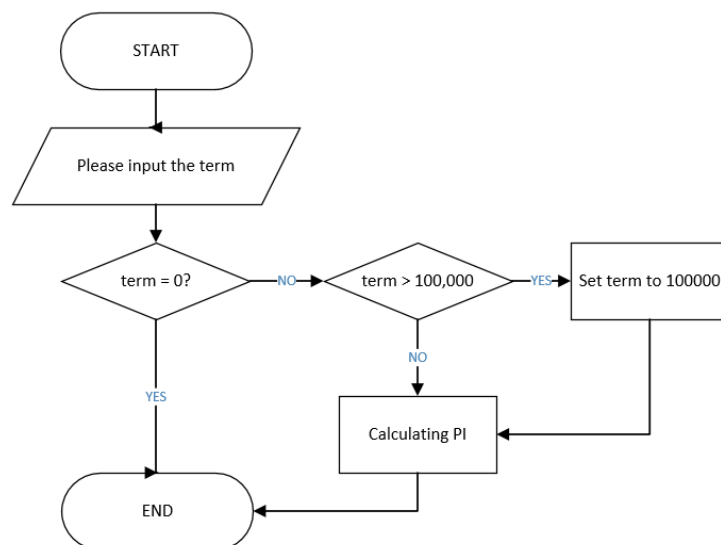


Figure 1

The figure above shows the general structure of my program. First, it tells the user to input term. Then the main program judges if the term is equal to zero. If term is zero, the whole program exits, all tasks end. If the term is larger than 1000,000, then the program will automatically set it to 1000,000. Then we finally get the legal term, which means ECX (the illiterate times) is determined. After that, we can start to use Taylor series to calculate PI.

How to translate the following equation (See 1) using assembly language is crucial

for our program, we will talk about the method further in the Methodology part. The FPU set plays an important role in achieving this goal.

$$\frac{PI}{4} = \sum_{i=1}^{term} (-1)^{2*i-1} \frac{1}{2*i-1} \quad (1)$$

```
term DWORD ?
dblOne REAL8 1.0
dblThree REAL8 2.0
dblFour REAL8 1.0
result REAL8 0.0
dblOut REAL8 ?
pc DWORD ?
```

Figure 2

The picture above shows the variables I defined in my program. PI is an infinite non-repeating decimals, so the type of all the operands in this program is REAL8. Of all the variables show in this picture, dblOut is the most important one, it acts as a role storing temporary results in the iteration. We store the final PI in result.

In terms of the mathematical unit of this program, we combine FPU and Stack to achieve our goal of simulating this equation. These following instructions are very important for this unit.

- finit: Initialize FPU after checking error conditions
- fld : load floating-point value to the Stack
- fstp: Store floating value and pop
- fmul fadd fdiv fsub: basic arithmetic instructions, the operands are floating-point
- mwrite: it's an internal function in Macros.inc, used for output message in the screen

Finally, I use internal function WriteFloat to output the final result to the screen.

3. Methodology (at least 300 words) [30%]

Word Count: _____ (must filled, or zero point)

The theory of stack is one basic prerequisite for the methodology of this program. The stack obeys the Last In First Out (LIFO) rule. The stack is an abstract data type that serves as a collection of elements [1]. It has two principal operations: PUSH adds an element to the stack while POP removes the most recently added elements. In this program, we use fld to load floating-point value to the stack and fstp to store the floating value and pop.

Here I would introduce to you how I design the arithmetic parts of my program:

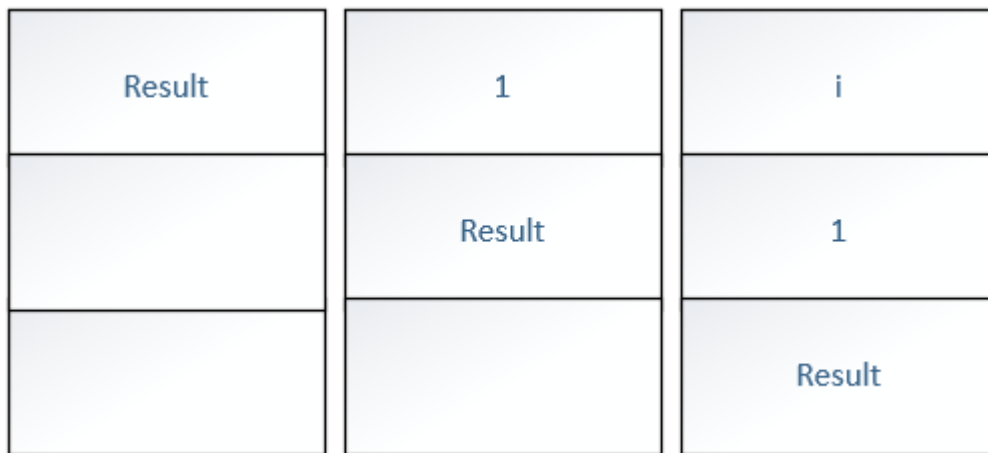


Figure 3

Figure 3 shows how I initialize my stack. There is no doubt that after term is determined, we should set a loop function to calculate PI. i is the iteration number (see equation 1), and the result is used to store the value of PI every time when the iteration ends.

We must ensure that $ST(0)$ is PC until $2*i-1$ is finished and then pop the present value of $ST(0)$ to `dblOut`. (See Figure 4)

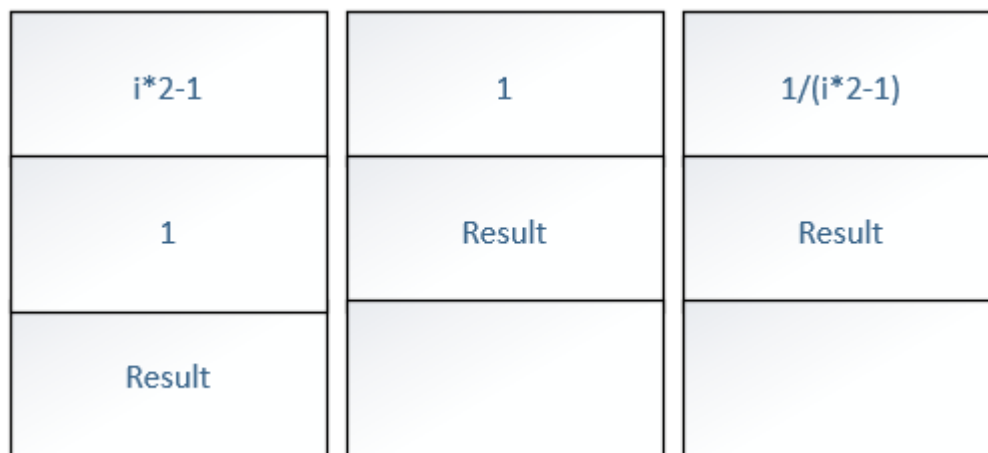


Figure 4

Then $ST(0)$ changes to 1, we apply the `fdiv` instruction to calculate $1/dblOut$ and pop the value to memory `dblOut`. At this time, there only exists result in the stack. We note that when i is prime then `dblOut` has a negative sign, and front sign of `dblOut` is positive when $2*i-1$ is even. In my program, PC records the state of i . When $PC == 1$, then subtract value of `dblOut` from result, otherwise, add the value of `dblOut` to result.

Before next iteration, we add 1 to i , and keep 1 at $ST(0)$, result at $ST(1)$. Finally when the times of iteration reaches term, the loop stops. The $ST(0)$ is result we want

and print it on the screen.

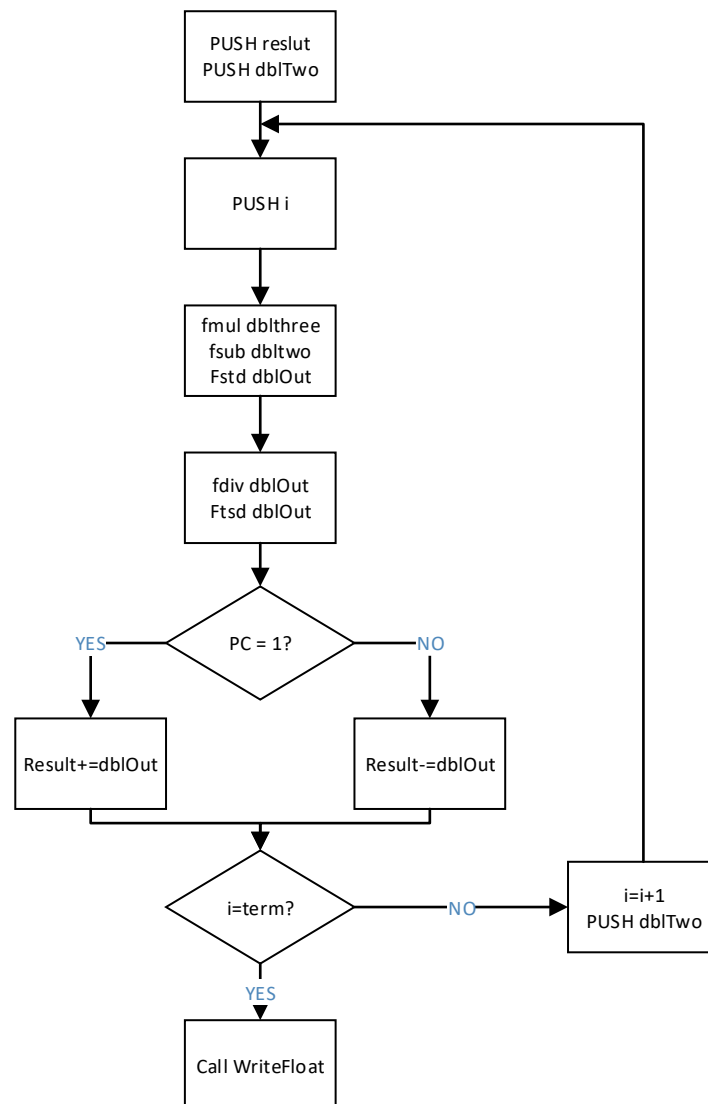


Figure 5

Figure 5 shows the structure of the arithmetic unit of my program. All together there are 35 lines of code for this part. But when you use High-level language (e.g. python), it only uses 8 lines to realize all the functions depicted in the Figure 5. We will use python as an important tool to help us analyse this assignment in Experiment part.

4. Experiments (at least 200 words) [20%]

Word Count: _____ (must filled, or zero point)

These pictures below shows the running states of my program. I use Visual Studio2015 and MSAM to build and debug my assembly program.

1、Enter the terms (Operation precision) and calculates the result:

```
Please input n which is the number of terms:1000000
Pi is:+3.1415826E+000
```

Figure 6

2、When the term is 0, then the whole program exists

```
Please input n which is the number of terms:0
请按任意键继续. . .
```

Figure 6

I use python to build this program and analyze the result when different terms are given. Here I draw a line chart (See Fig 7) where x-axis is term and y-axis is the result, it clearly shows that as the term increases, the result gets closer to value PI .

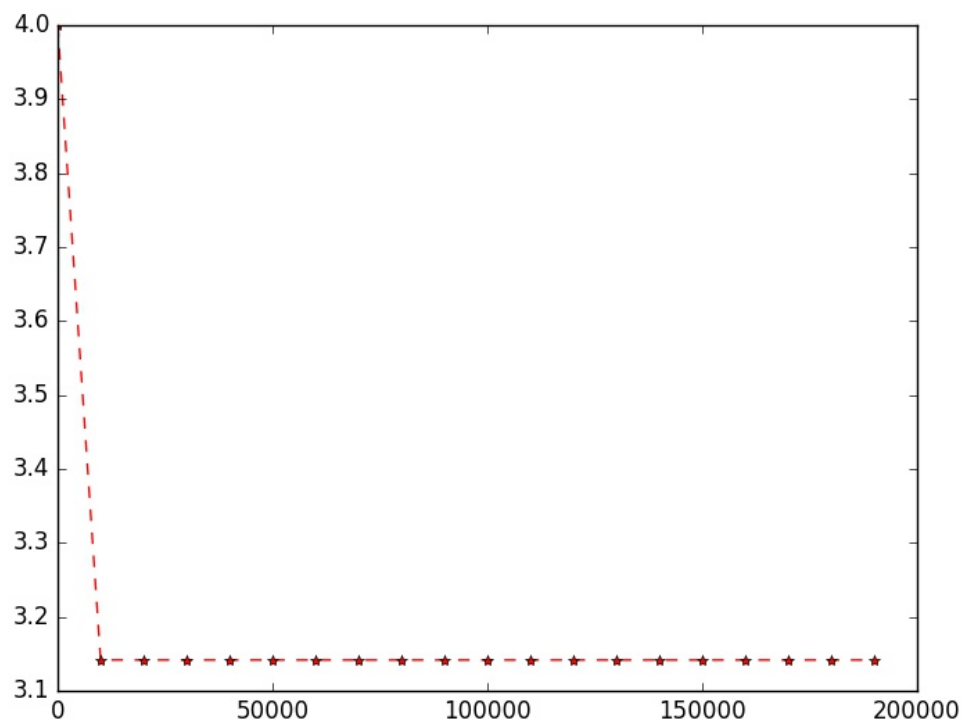


Figure 7

Term	Result
100	+3.1315929E+000
200	+3.1365926E+000

300	+3.1382593E+000
500	+3.1390926E+000
1000	+3.1395926E+000
10000	+3.1414926E+000
100000	+3.1415826E+000

Table 1

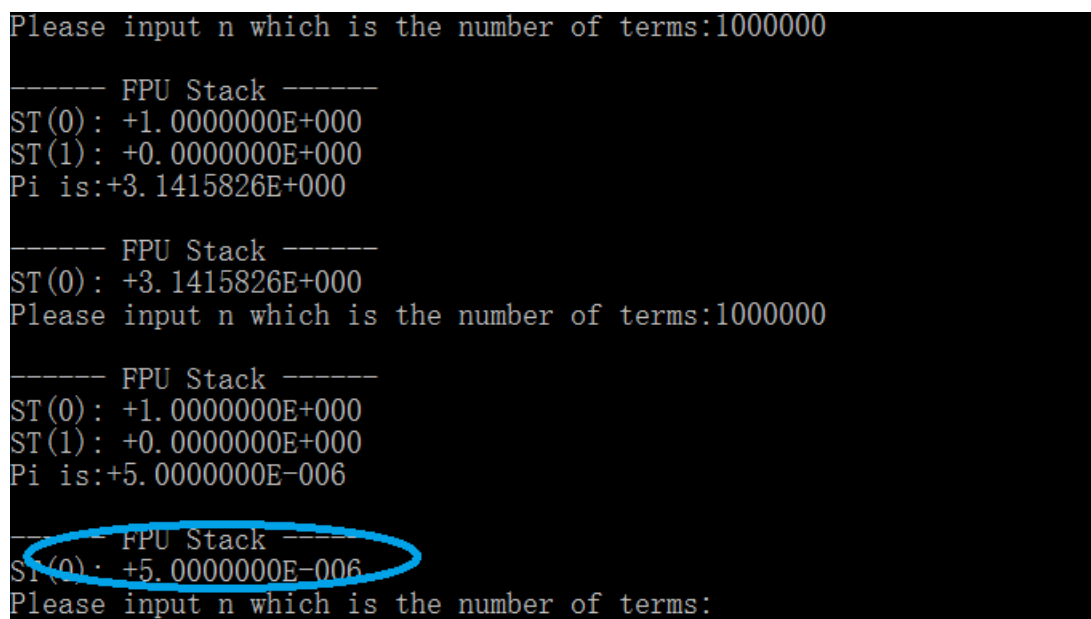
Comparing to python, assembly language is much faster. When term is 1000000, it takes python about 0.5s to finish calculation.



```
3.14159165359
read: 0.520366 s
```

Figure 8

I want to make my program more robust and humanized, so I add an additional loop to my program and reconstruct its frame to make it easier to read. However, I met a lot of troubles when debug my new program, and the problem below is a typical one:



```
Please input n which is the number of terms:1000000

----- FPU Stack -----
ST(0): +1.0000000E+000
ST(1): +0.0000000E+000
Pi is:+3.1415826E+000

----- FPU Stack -----
ST(0): +3.1415826E+000
Please input n which is the number of terms:1000000

----- FPU Stack -----
ST(0): +1.0000000E+000
ST(1): +0.0000000E+000
Pi is:+5.0000000E-006

----- FPU Stack -----
ST(0): +5.0000000E-006
Please input n which is the number of terms:
```

Figure 9

As is can be seen from Figure 9 that the output result in different with the same input. I tried very hard to debug it and find that I did not initialize PC and i after one iteration is done. The result is that the result gets smaller and you may get a negative result. Figure 10 confirms my assumption: i is not initialized at all.

```

Please input n which is the number of terms:1000000
+1.0000000E+000
----- FPU Stack -----
ST(0): +1.0000000E+000
ST(1): +0.0000000E+000
Pi is:+3.1415826E+000

----- FPU Stack -----
ST(0): +3.1415826E+000
Please input n which is the number of terms:1000000
+1.0000100E+005
----- FPU Stack -----
ST(0): +1.0000000E+000
ST(1): +0.0000000E+000
Pi is:+5.0000000E-006

----- FPU Stack -----
ST(0): +5.0000000E-006
Please input n which is the number of terms:请按任意键继续. . .

```

Figure 10

After all the bugs are eliminated, my new program runs like this (See Figure 11).

```

*****
This is Mr.Bean's first homework
*****
Please input n which is the number of terms:1000000

Pi is:+3.1415826E+000
Please input n which is the number of terms:1

Pi is:+4.0000000E+000
Please input n which is the number of terms:2

Pi is:+2.6666666E+000
Please input n which is the number of terms:3

Pi is:+3.4666666E+000
Please input n which is the number of terms:4

Pi is:+2.8952381E+000
Please input n which is the number of terms:0
请按任意键继续. . .

```

Figure 11

5. Conclusion (at least 150 words) [15%]

Word Count:_____ (must filled, or zero point)

Assembly language is a low-level programming language for computer. It has the characteristic of high efficiency, space-saving, etc. However, in order to realize one

specific task, assembly language may have to use more instructions to perform the same logic than one high-level language does and this makes assembly language program hard to read. So always remember to use notations when you make ASM program.

After finishing this assignment, not only do I become more familiar with some basic knowledge of ASM, such as the way to set the building environment, FPU set, Stack theory, but also I cultivate my sense of active learning and the ability to express myself. We do not know how to use FPU set before this task is assigned, however, I learn it from my textbook and Internet by myself, after trying many times, finally I grasp this knowledge. Writing a report using English is also a challenge for me, how to better express my methodology in the report is not easy, this process is very dull, but I really benefit a lot after finishing it.