**Assembly language and system programming**

Assignment 01

Student Name: 楊家安

Student ID: 0516021

Student email address: gxxygxxy@gmail.com

**[10%] Introduction [ at least 100 words]**

**WORD COUNT:\_\_\_\_\_\_\_130\_\_\_\_\_\_ [ Must be filled or zero score]**

This is homework 01.

In the menu, there are six options in this program that the user can choose from. After showing the result of each option, user can press any key to go back to the menu.

The first option is to show the colorful frames in random colors, which won’t be black. Also, two consecutive frames won’t have the same color. The width is 84 and the height is 24.

The second option is to sum up several signed integers.

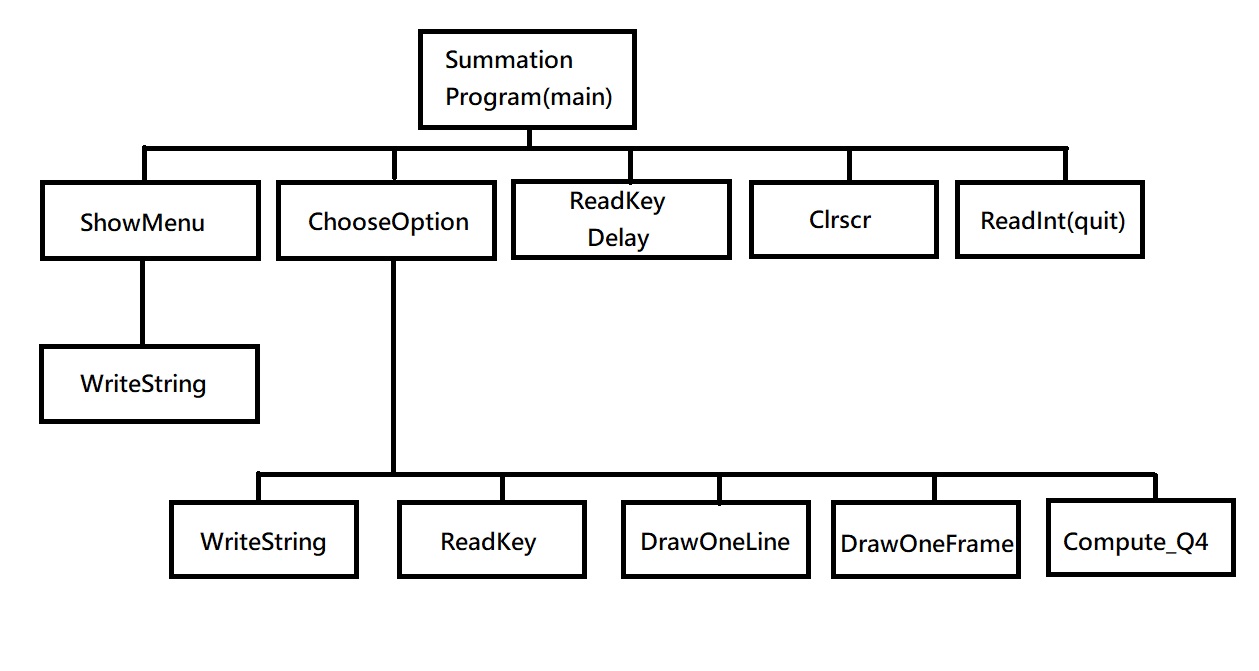
The third option is to sum up several unsigned integers.

The fourth option is to compute the sin(x). The user is required to input the x and the terms.

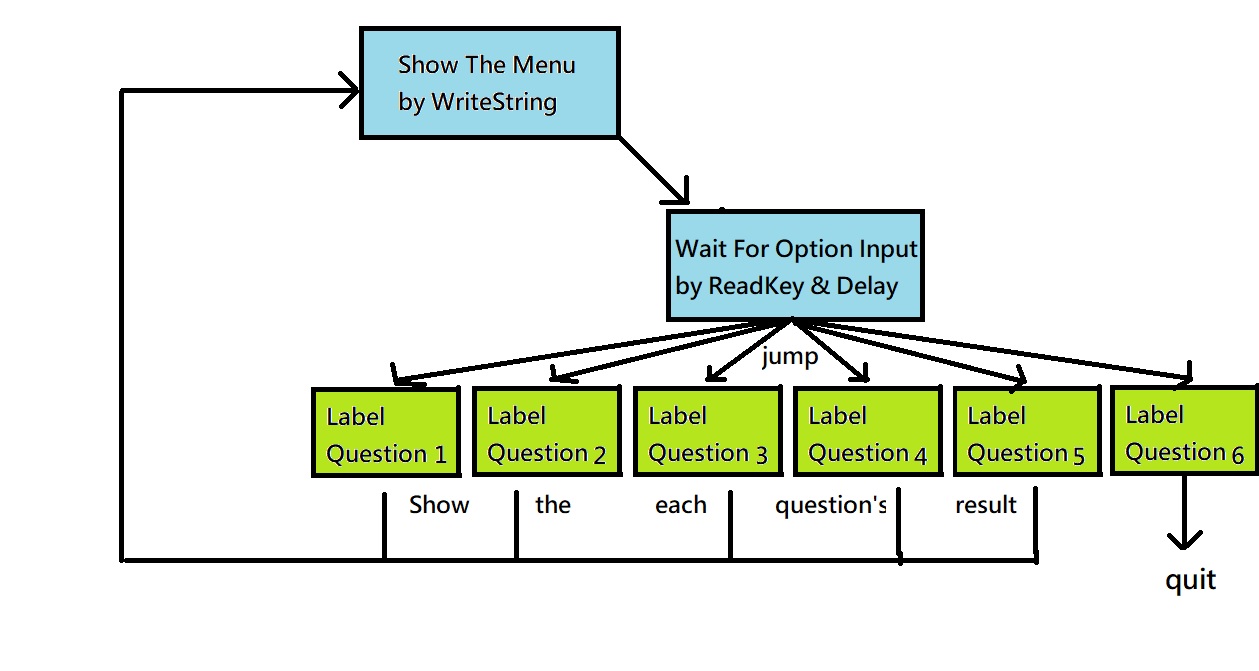
The fifth option is to show the information of the student.

The sixth option is to quit the program.

**[10%] System Chart** [ at least 10 components]



[**10%] Flow Chart**



**[10%] System Architecture** [**at least 100 words]**

**WORD COUNT:\_\_\_\_\_\_\_117\_\_\_\_\_\_ [ Must be filled or zero score]**

In the main, there are procedures: ShowMenu, ChooseOption, Delay, ReadKey, SetTextColor, Clrscr and ReadInt. For instructions: mov, jz, loop.

In the procedure ShowMenu, there are procedures: WriteString and Crlf. For instructions: mov.

In the procedure ChooseOption, there are procedures: Delay, ReadKey, Clrscr, DrawOneFrame, GotoXY, ReadDec, ReadInt, ReadFloat, WriteFloat, WriteDec, WriteInt, Crlf, ComputeQ4. For instructions: mov, jz, je, inc, sub, cmp, jmp, mWrite, add, finit, fst, fld.

In the procedure DrawOneFrame, there are procedures: Delay, Randomize, Random32, SetTextcolor, DrawOneLine. For instructions: mov, and, cmp, je, xor, jle.

In the procedure DrawOneLine, there are procedures: WriteChar and GotoXY. For instructions: mov, and, loop.

In the procedure Compute\_Q4, there are no procedures, but instructions: fimul, fmul, fidiv, fadd, fstp, fst.

**[30%] The approach [ at least 300 words]**

**WORD COUNT:\_\_\_\_\_\_\_342\_\_\_\_\_\_ [ Must be filled or zero score]**

In the main, I first call the procedure ShowMenu to show the menu by WriteString. Then I call the procedure ChooseOption, to wait for the input of the option by ReadKey and Delay. After the input, according the input to jump to the compatible label to show the detail of the option, by using cmp and je. After showing the result, return to the main and wait for any key to show the menu again by using ReadKey and Delay.

Option one:

Clear the screen first. Draw a frame which is not black by using bitmask (black is 0000????). We also need to store the color to the variable LastFrameColor. And then increase dl, dh and draw another frame.(If the color of the new frame is same as the old frame i.e. LastFrameColor or it’s black, then do the loop re-randomize the new color till not the same)

Option Two:

Initialize the answer to 0. ReadDec to input the number of signed integers. Move the input to ecx and then loop: ReadInt and add eax to answer.

Option Three:

Initialize the answer to 0. ReadDec to input the number of signed integers. Move the input to ecx and then loop: ReadDec and add eax to answer.

Option Four:

finit first , to clean the trash in the ST. This is very important.

* ST(0) used to store the term, ST(1) used to store the answer.

Initialize the ST(0) and ST(1) to x first. Initialize the variable denominator to 1. Then do the loop n times.

Within each loop, multiply the -1 and x^2 to ST(0), and div the two accumulative denominator(1! 3! 5! 7!...). Thus we get a new term in the Taylor Series. Finally add ST(0) to ST(1).

The answer will store in the ST(1) at the end. Pop ST(0) to trash and just get the answer in ST(0). Use fst to store the answer in the variable Question4Ans.

Option Five:

mWrite the information of the student, and then return to the main procedure.

Option Six:

Directly use the instruction quit.

**[20%] Discussion/Experiments [ at least 200 words]**

**WORD COUNT:\_\_\_\_\_\_288\_\_\_\_\_\_\_ [ Must be filled or zero score]**

The first time I encountered the trouble was to choose the option. I used the procedure ReadKey but it was not work. After getting information with the Internet, I realized the way I used the ReadKey is wrong. If you doesn’t input anything, the ReadKey still works immediately. To solve this problem, I called the Delay on it with loop and the jz to judge if there is any input key. If you doesn’t input, the loop and the jz then works.

The next small trouble is how to avoid to show the black frame in option one. To solve this, I use the bitmask. Because the black frame in al must be 0000????. I can easily solve this by AND 11110000b then check if it’s equal to 00000000b. If so, then re-randomize.

Also in the option one, the consecutive frame must be different color. So I have to memorize the last printed frame color, and if the randomized color is the same as the last printed one, then re-randomize.

In Option four, to deal with the float number, I also use instructions such as fmul, fdiv….I doesn’t know the detail of these instructions so I also google them a lot, and realize the difference between them.

The hardest is how to compute the Taylor series. Each terms is multiplied by -1 and x^2 and divided by two more accumulative numbers. I think store the answer in ST(1) and store the term in ST(1) is easy to implement because we can easily do operands on ST(0)

The homework also enhances the thing I learn from the lesson. Such as the eax, ax, ah, al and a few instructions. The most challenging part is the option one and four.

**[10%] Conclusion [ at least 100 words]**

**WORD COUNT:\_\_\_\_\_\_\_166\_\_\_\_\_\_ [ Must be filled or zero score]**

This homework makes me learn a lot. In the menu, I learn how to call ReadKey with loop and judge by je.

In option one, I learn how to randomize and GotoXY, and implement with bitmask.

In option two and three, it’s relatively easy by using ReadDec, loop, and add.

In option four, I learn how to deal with float numbers in ST. also, help me recall the Taylor Series in Calculus.

Coding in assembly language is interesting, because I only coding in C/C++/Python before. They are very different! In assembly language, you have to code instructions line-by-line, whereas in high-level language, you can code one line, which means several instructions.

In other lessons I take, I also learn the MIPS. Although one is in x86, the other is in x64. With different instructions, many concepts are the same, e.g. the registers and the jump.

The homework is also challenging and interesting. It’s helpful in learning assembly language. Hopefully I could learn more in the assembly.