**Hanoi University of Science and Technology  
School of Information and Communication Technology**



**PROJECT REPORT**

**Project name: Integral Calculator**

***Subject: Assembly Language and Computer Architecture Lab***

**Instructor: Dr. Do Cong Thuan**

**Group 3:**

|  |  |  |
| --- | --- | --- |
| **Student ID** | **Student name** | **Class** |
| 20194787 | **Tran Lam** |  |
| 20194849 | **Chu Thach Thao** |  |
| 20184238 | **Nguyen Thi Minh Chau** | **ICT 01 – K63** |

**Hanoi, 2021**

TABLE OF CONTENTS

# **PART I: PROJECT OVERVIEW**

Calculation is one of the main objectives of calculators and computers. With its power of performing multiple operations quickly at the same time, these devices are designed to perform difficult tasks. All the instructions written in high-level programming languages will be converted into assembly code, then binary code for processor to understand and behave accordingly.

However, there are differences in programming in modern programming languages and assembly languages. The actual instructions needed in order to perform one task will be different when being determined in human-friendly code and machine code. Although current widely-used programming language provide us convenience when working, there are mechanisms and other ways of ordering smaller element of code so that the program can run more effectively.

To ultilize the power of these machines, we analyze the underlying concepts, as well as the basis of a program, at the machine level, and apply these knowledge to reality to create a program that performs tasks as desired.

In this project, we implements different functions to estimate the integral, given some boundaries. Integration is one of the most basic operations populary used in different fields, to determine different values for particular purpose. Although this mathematical problem can be performed with pens and papers, the amount of work is huge and it can be reduced with the help of computer and a proper calculate strategy.

# **PART II: PROJECT IMPLEMENTATION**

## **1. Insights**

Intergration is a mathematical problem used to detemine the area or volume of object that has complex shape. This is one of the most basic operation that most engineer in all fields need to conquer in order to design proper solution for their work. Integration can be done in various ways, and there are also many different techniques to solves different kinds of integrals. Even one basic problem of integration has multiple method to implement.

The core idea of integration is to divide the surface or volume in to relatively small segment, each of which can be considered as a basic trapezoid or regtangle, which is easy for calculation, to determine the value of them, then sum up them all to get the final value.

This estimation procedure has small errors that can be accepted. The smaller the segment is, the more segments that the shape is divided into, the more correct the operation is.

In this project, we need to determine the area bounded by the lines:

To do this, we use 3 techniques:

* Naive: using the basic idea of integration
* Newton-Leibniz: the integral formula that has been taught widely from middle school
* Square: transform the function to integrate

For each technique, we have a set of test cases and compare the result with which of the integral calculator (at webpage….)

## **2. Naive method**

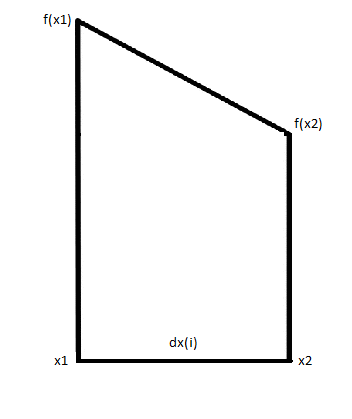
### *a. Overview*

Naïve method we’re considering here is based on Trapezoidal Rule. To calculate the integration the total area is divided into small trapezoids. Sum of area of all the trapezoids is the answer of our problem.

The Naïve method is easy to understand to human and also easy to implement in the computer. The precision of this method depends on the number of trapezoids we divide. More trapezoids means higher precision.

### *b. Implementation*

The implementation of this method is pretty straight-forward. Consider the bounds are a and b   
(a < b) and we divide the region into n parts.



The first trapezoid has

The trapezoid has

Finding and for each trapezoid is handled by a loop different for each way.

For even division

For sloppily small

Finding is handled by procedure calculate\_f

Finding the area of the trapezoid is handled by procedure area

The final answer is sum of all area of the trapezoids (1) to (n)

This rule is used for approximating the definite integrals where it uses the linear approximations of the functions.

In this implementation, we’ll experience Naïve method in three ways: even division (height of all trapezoids are the same) integrate from left to right and right to left, sloppily small (height of trapezoids increase from left to right) integrate from right to left.

### *c. Result*

* Case 1: a = 0, b = 2, n = 10

Even division - left to right: 4.426463558338128

Even division - right to left: 4.42646352627467

Sloppily small - right to left: 4.549573503465402

* Case 2: a = 0, b = 4, n = 20

Even division - left to right: 5.302901700779358

Even division - right to left: 5.302901924703356

Sloppily small - right to left: 5.584574132449071

* Case 3: a = 0, b = 100, n = 10000

Even division - left to right: 6.24318662919517

Even division - right to left: 6.243186904586415

Sloppily small - right to left: NaN

* Case 4: a = 0, b = 1
  + n = 10

Even division - left to right: 3.139926017069971

Even division - right to left: 3.1399259908390924

Sloppily small - right to left: 3.1492665141208205

* + n = 1000

Even division - left to right: 3.141592492903594

Even division - right to left: 3.141592481345689

Sloppily small - right to left: NaN

* + n = 10000

Even division - left to right: 3.141592651694072

Even division - right to left: 3.1415926502744327

Sloppily small - right to left: NaN

* Case 5: a = 0, b = 1e9
  + n = 1e5

Even division - left to right: 20000.000457950562

Even division - right to left: 2364736.0000033793

Sloppily small - right to left: NaN

* + n = 1e6

Even division - left to right: 2000.00648706424

Even division - right to left: -1.52127519538686E7

Sloppily small - right to left: NaN

### *d. Comments*

The precision of this method is not high because there are many steps that errors can occur.

The first problem is the number of trapezoids we divide n. Mathematically, larger n results in more precise answer. However, it’s difficult to calculate with big n in computers because the error will also be big and the runtime is not efficient.

Second is the step of finding . For sloppily small the error is huge because with big n we have is very big and calculating this in computer (shift 1 bit to the left) we’ll lose a lot of bits, and the division is 99% not correct because . For even division the error is smaller but can still occur.

Third problem is calculating We have . With big value of we’ll have big and normally we’ll see that and the division will have error.

Errors in other steps of the program will pile up from the problem of because we use in calculating the area of the trapezoid. And since we have many components in the sum, larger n means larger error.

To sum up, in theory larger number of trapezoids will give more precise answer but that is not the case in implementing this method on computer.

## **3. Newton-Leibniz method**

### *a. Overview*

Newton – Lebniz is a famous formula to calculate integral that is taught to student from middle schools. Suppose that one area is bounded by 2 functions: and , and its x-coordinates of the conjuntions are a and b (from left to right, repsectively), then the surface can be calculated by:

To determine the area bounded by x=0, y=0, y=a and f(x), we calculate

In this project, we need to calculate

However, computers do not understand how to do integrals – they can only perform basic operations such as addition, substraction and shift. Therefore, we need to convert the formula into instructions corresponding to operations that computers can do.

Note that, every function can be approximtely written into a series of numbers. In this case, an arctan function can be rewritten as

Therefore, to implement the function on computer, we can convert it into sum of a (large) number of terms, each of which is determined by the formula above.

However, since the representation of the floating number in computers is limitted, there will be precision loss in calculations, as well as the final result

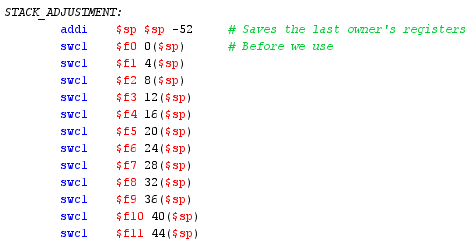
### *b. Implementation*

Based on the above idea, the program for determining the area under the given curve is implemented to work with 2 Taylor series, and the final result is the difference of them.

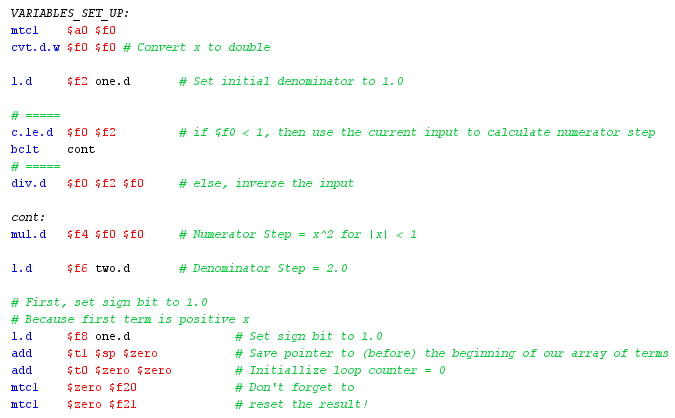
Using basic loops for calculation powers and terms of series, for each loop, we will caculate corresponding exponent number and denominator of the terms, before adding it to the final result of the series calculation. Therefore, we will need 2 main loops in order to calculate 2 series, and before subtracting them and get the final result.

The naïve approach to this program is to use a sub-function to calculate the power for nominator of each terms. This will lead the complexity of the program be since we need an amount of loop for determining the exponent, and loop this process for n times for n terms

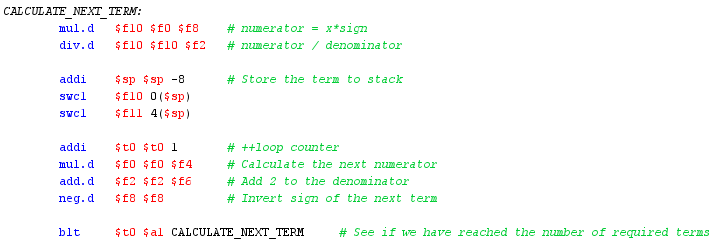
In this program, we use stack of f-registers to save information, since each term is represented in floating-point format.



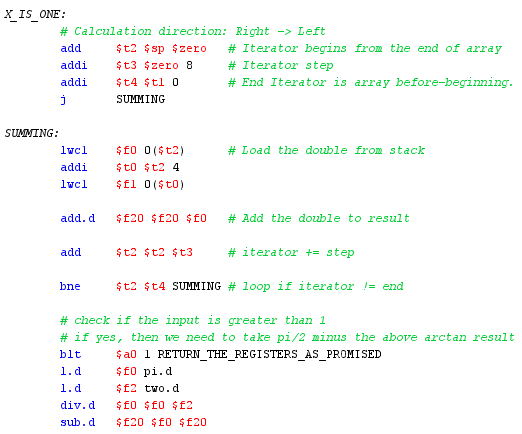
Before getting into determining value of each term, we initialize the stack and variables. Each floating point will be stored in a pair of registers, and we need to decrease the stack enough to appropiately save the pointers.



In this program, we want to ultilize the precicion of double type, therefore, initial values must be converted to double for calculation. Initial denominator is 1. Step for denominator is 2, and for numerator is . By doing this, we can avoid using a loop for calculating power value for the numerator, thus reduce the complexity of the program from to . One register is used to point to the (before) start position of the array.

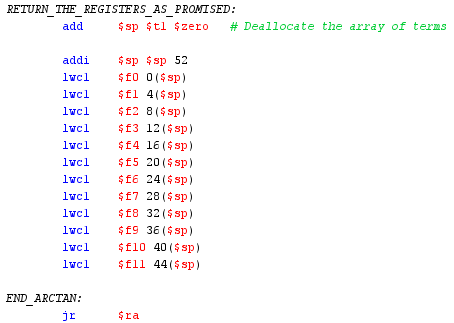


After having initialize the values for calculation, one loop is used to determine value of each term. Each of the result will be put on stack by the 3 instructions from 3rd to 5th in the image above.

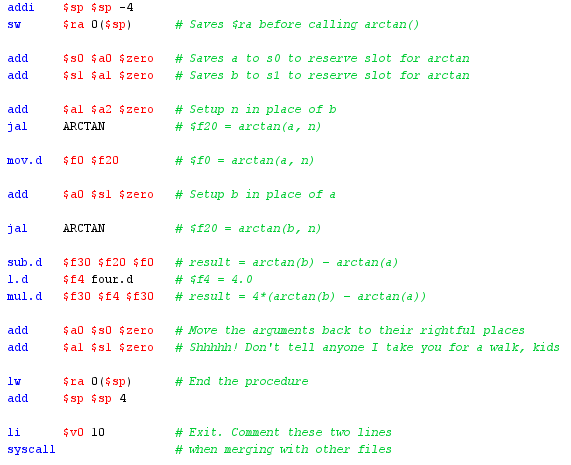


Since the series will have each term smaller and smaller, to prevent precision loss, we take advantages of the stack pointers and start adding each term of the series from right to left.

After having finished calculating each serires, the stack is released to free memory and return space for further operations.



After having all the values of the 2 series, we do the subtraction and save the value in the destination register, return the stack pointer value and terminate the program.



### *c. Result*

* Case 1: a = 0, b = 2, n = 10 (Example test case)

Result: 4.428594945143271

Deviation: 7.3966909.10^-8

* Case 2: a = 0, b = 4, n = 20

Result: 5.303270654672129

Deviation: approximately 0

* Case 3: a = 0, b = 100, n = 10000

Result: 6.243186640432925

Deviation: approximately 0

* Case 4: a = 0, b = 1e9, n = 1e5

Result: 6.283185303179586

Deviation: 0

* Case 5: a = 0, b = 1e9, n = 1e6

Result: Did not pass

### *d. Comments*

The series implementation gives precise result because the operations used in calculation can be done easily with negligible errors. The correctness increases when the step (n) is larger. However, since the resources is limited, when n is too large, we encounter stack overflow and the program cannot reach the end.

In reality, series is also used in many applications and also produce result with acceptable errors.

# ***APPENDIX I: TABLE OF FIGURES***

# ***APPENDIX II: SET OF TEST CASES***

1. a = 0, b = 2, n = 10 (Example test case)

integral-calculator: 4.428594871176362

2. a = 0, b = 4, n = 20 (Project requirement's maximum constraint)

integral-calculator: 5.30327065467213

3. a = 0, b = 100, n = 10000

integral-calculator: 6.243186640432926

4. a = 0, b = 1e9, n = 1e5

5. a = 0, b = 1e9, n = 1e6

integral-calculator: 6.283185303179586

# ***APPENDIX III: REFERENCES***

# ***APPENDIX IV: GROUP ASSIGNMENT***

|  |  |
| --- | --- |
| ***Team member*** | ***Assignment*** |
| Trần Lâm | Team leader  Project initialization  In charge of Square method  Check & integrate final solution |
| Chu Thạch Thảo | In charge of Naïve method |
| Nguyễn Thị Minh Châu | In charge of Newton – Leibniz method  Prepare final report & presentation |