INDIAN INSTITUTE OF TECHNOLOGY, DELHI

ASSIGNMENT DESIGN

Problem Statement

Write a MIPS Assembly Program for obtaining the area under a curve formed by joining successive points by a straight line.

APAR AHUJA | ENTRY NO. 2019CS10465

ARNAV TULI | ENTRY NO. 2019CS10424

Course - COL216 | Prof. Preeti Ranjan Panda

Compiled on February 22, 2021

COL216 - Assignment 1

Arnav Tuli | Apar Ahuja

February 22, 2021

1 Approach and Design

1.1 Input - Output

- Source Code: **assignment_1.asm** takes input from the **keyboard** and prints the output on **console**.
- Tester Code: **tester.asm** takes input from the **Test_Input.txt** and prints output on **console**.
- Use **double precision** as area calculations showed significant errors during single precision testing
- Also, double precision registers use 64 bits and have a positive number range from $\mathbf{10^{-306}}$ **to** $\mathbf{10^{306}}$.
- Therefore, it is safe to assume that there will **not be any overflow** during area calculations.
- Using double precision registers also make our program flexible as it can now work with real inputs as well.

1.2 Algorithm

- Initialize **Counter** to 1 and **Area_Till_Now** to 0
- Read and store user input **n** = number of points
- **First Input:** Read user input as (prevX, prevY) := (x_1, y_1)
- Increment **Counter** by 1
- **loop:**

. - If **Counter** > **n** then **Display Area**, else read user input as (currX, currY) := (x_i, y_i)
. - Calculate the area under the curve between prev and curr and add it to **Area_Till_Now**
. - *(Area calculation and derivation is attached at the end)*
. - Update prevX and prevY to currX and currY. **(prevX, prevY) := (currX, currY)**
. - Increment **Counter** by 1.
. - **Jump** back to **loop**

- **Display Area** prints the area on the console
- **Loop Invariant:** Before iteration **i**: **Area_Till_Now** = area enclosed by first **i** points. $\mathbf{1 \leq i \leq n}$.

1.3 Design

1.3.1 Register

Integer Registers:
v0: used for making different syscalls, and also storing user inputs
a0: used in making syscalls (outputting strings and integers/double)
t0: Stores 'n', the total number of points
t1: Stores 'counter' ($1 \leq$ counter $\leq$ n + 1)

Floating Point Registers (Double Precision):
f0: used for storing user input when syscalls are made | also stores currY (y_2) while calculating area
f6: the constant HALF (0.5)
f4: value of area calculated till now
f8: stores prevX (x_1) while calculating area
f10: stores prevY (y_1) while calculating area
f12: stores currX (x_1) while calculating area | also used to make syscalls to display area calculated on console

f14: stores the product $y_1 * y_2$ while calculating area
f16: stores $x_2 - x_1$ while calculating area
f18: stores either $y_1 + y_2$ or just y_1^2, depending on location of points
f20: stores either $| 0.5 * (x_2 - x_1) * (y_1 + y_2) |$ or just y_2^2 depending on location of points
(* used only when points are on opposite side of X-axis *)
f22: stores $y_1^2 + y_2^2$
f26: stores $| 0.5 * (x_2 - x_1) * (y_1^2 + y_2^2) / (y_2 - y_1) |$ also used while comparing $y_1 * y_2$ with 0.0
f28: stores $y_2 - y_1$

1.3.2 Main Memory

ASCII:
msg_input: .asciiz "Enter the number of points (n): "
error_zero: .asciiz "Error: Number of points is zero. Area under the curve is not defined."
error_invalid: .asciiz "Error: Number of points cannot be negative."
error_notSorted: .asciiz "Error: Points not sorted w.r.t X-coordinate."
newline: .asciiz "\n"
msg_point: .asciiz "\nPoint "
msg_Xcod: .asciiz "Enter X coordinate: "
msg_Ycod: .asciiz "Enter Y coordinate: "
msg_separator: .asciiz "\n\n____________________\n\n"
msg_total: .asciiz " \n Total area enclosed by the line plot and X-axis is: "

DOUBLE:
prevX: .double 0.0
prevY: .double 0.0
currX: .double 0.0
currY: .double 0.0

1.4 Raising Errors

- if **n = 0** we raise *Error: Number of points is zero. Area under the curve is not defined.*
- if **n < 0** we raise *Error: Number of points cannot be negative.*
- if input points are **not sorted** we raise *Error: Points not sorted w.r.t X-coordinate.*

2 Testing Strategy

- Total of **420** test cases were generated and tested against as a part our extensive testing strategy
- use *TestCaseGenerator.py* to generate **randomized** test case files with correct output.
- *tester.asm* reads file and prints output on console. We then copy it into a text file and run *Checker.py*.
- *Checker.py* calculates the difference upto 12 decimal places and stores it in "Difference_12_decimal_places.txt"
- We store count of cases with 0 difference upto 12 decimals and total number of cases to calculate **accuracy**.
- Types of Test Cases Used:
. - Test Cases with **10**, **50**, **1000** and **10000** randomized points with varying coordinate ranges
. covering all possible cases (**positive Y**, **negative Y**, **mixed Y** and **stacked X**)
. - **Corner cases** for n $\leq$ 0 or unsorted input or single point input
. - Manually generated cases include only positive Y, only negative Y, mixed positive-negative Y
. and input stacked w.r.t X types of cases. (for e.g. - [(1,2), (1,3), (1,4), (5,-1)])

3 Result

We achieved a **100%** accuracy across all our test cases, with **0** difference in all outputs upto **12** decimal places.

Console Interface -

Console

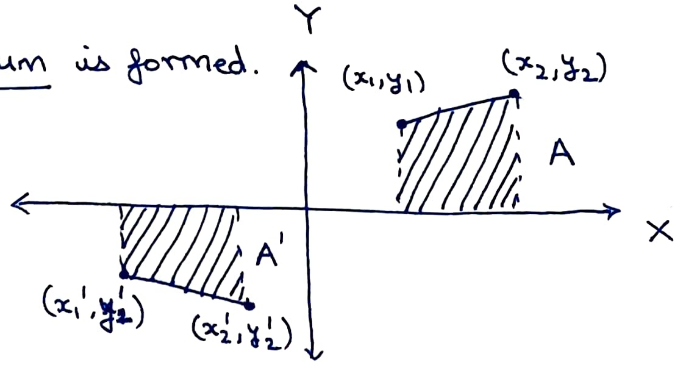
```
Enter the number of points (n): 5

Point 1
Enter X coordinate: 1
Enter Y coordinate: 1

Point 2
Enter X coordinate: 3
Enter Y coordinate: 4

Point 3
Enter X coordinate: 5
Enter Y coordinate: 3

Point 4
Enter X coordinate: 6
Enter Y coordinate: 7

Point 5
Enter X coordinate: 9
Enter Y coordinate: 5

Total area enclosed by the line plot and X-axis is: 35

________________________________________________________________

Enter the number of points (n): 0
Error: Number of points is zero. Area under the curve is not defined.

________________________________________________________________

Enter the number of points (n): -1
Error: Number of points cannot be negative.

________________________________________________________________

Enter the number of points (n): 1

Point 1
Enter X coordinate: 2
Enter Y coordinate: 3

Total area enclosed by the line plot and X-axis is: 0

________________________________________________________________

Enter the number of points (n): 3

Point 1
Enter X coordinate: 1
Enter Y coordinate: 2

Point 2
Enter X coordinate: 3
Enter Y coordinate: 4

Point 3
Enter X coordinate: 2
Enter Y coordinate: 5
Error: Points not sorted w.r.t X-coordinate.

________________________________________________________________
```

AREA FORMED BETWEEN A LINE SEGMENT (joining (x_1, y_1) and (x_2, y_2)) AND X-AXIS (Formula)

Case I : $y_1 y_2 \geq 0$

In this case, both ends of the line segment lie on the same side of the x-axis

Hence, a trapezium is formed.



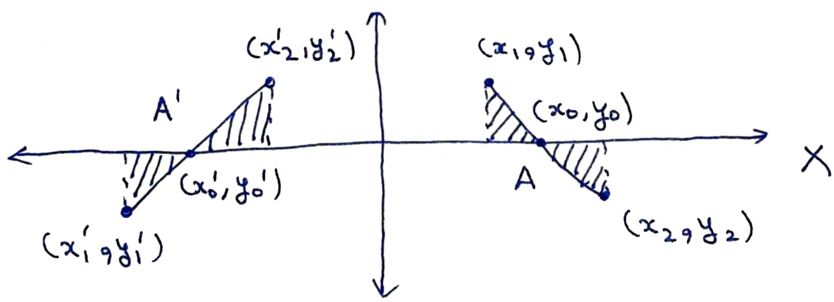
From area of trapezium, I get

$$A = \frac{1}{2}(y_1 + y_2)(x_2 - x_1) \quad \text{and} \quad A' = \frac{1}{2}(-y_1' - y_2')(x_2 - x_1)$$

So, in general, area formed = $\underbrace{|0.5 * (y_1 + y_2)(x_2 - x_1)|}_{\text{Formula used in algorithm (code)}}$

Case II : $y_1 y_2 < 0$

In this case, ends of the line segment lie on opposite sides of the x-axis. Hence, two triangles are formed.



Finding (x_0, y_0):

(x_0, y_0) lies in the interior of the segment $\Rightarrow$

$$x_0 = \frac{\lambda x_2 + x_1}{\lambda + 1} \quad \text{and} \quad y_0 = \frac{\lambda y_2 + y_1}{\lambda + 1}, \quad \lambda > 0$$

and y_0 lies on X-axis $\Rightarrow y_0 = 0 \Rightarrow \boxed{\lambda = -y_1 / y_2}$

So, $x_0 = \dfrac{y_2 x_1 - y_1 x_2}{y_2 - y_1}$ and $y_0 = 0$

Now, area $A =$ sum of two shaded triangles

$$= \frac{1}{2}(x_0 - x_1)\cdot y_1 + \frac{1}{2}(x_2 - x_0)(-y_2)$$

$$= \frac{1}{2}\, y_1 \frac{y_1 (x_1 - x_2)}{(y_2 - y_1)} + \frac{1}{2} \frac{(-y_2)\, y_2\, (x_2 - x_1)}{(y_2 - y_1)}$$

$$= -\frac{1}{2} \frac{(y_1^2 + y_2^2)(x_1 - x_2)}{(y_1 - y_2)} = \frac{1}{2} \frac{(y_1^2 + y_2^2)(x_2 - x_1)}{y_1 - y_2}$$

Similarly, $A' =$ sum of two shaded triangles

$$= \frac{1}{2} \frac{(y_1'^2 + y_2'^2)(x_2' - x_1')}{y_2' - y_1'}$$

In either case, area formed $= \underbrace{\left| \dfrac{0.5 * (y_1^2 + y_2^2)(x_2 - x_1)}{(y_2 - y_1)} \right|}_{\text{Formula used in algorithm (code)}}$