0.1 Approach

We Made an outline of the task we have to complete and broke it down into stages. In this case we prepared the following stages.

1. Make a procedure (input) to take coordinates as input. Also display instructions to the user while asking for input.

2. Make a procedure (area) that takes 2 points and returns the area of the trapeziod formed by those points. The area is calculated using single precision floating point registers. This procedure assumes that the input points are on the same side of the X axis.

3. In the main procedure, using the procedure input, accept coordinates from the user. Call the procedure area using the coordinates received from the user. If two successive coordinates (x_1, y_1) and (x_2, y_2) are on different sides of the X-axis (which we determine by checking the sign of the product $y_1 y_2$), we break the area into 2 triangles, one lying on each side of the X-axis, and call the procedure area on each separately.

4. An accumulator keeps track of the sum of all areas calculated, and after all the points have been processed, stores the area, which is to be displayed to the user.

0.2 Formulae used

- Area of trapeziod formed by (x_1, y_1) and (x_2, y_2) when both points are above the X-axis is given by

$$Area = (y_2 - y_1) * (x_1 + x_2)/2$$

.

- When both of the points are below X-axis then the negative of the above expression gives the absolute area.

- In the third case, we have that one point is above the X-axis and one is below. In this case, we introduce a third point, M on the X-axis where the line joining points A and B meet the X-axis. Then we can calculate the absolute area covered by A,B to be the sum of the areas of triangles formed by A,M and M,B. The coordinates of M are given by -

$$M = (x_1 - y_1.(x_2 - x_1)/(y_2 - y_1), 0)$$

0.3 User Interface

On running the program, we are first prompted to enter the number of points. Then on each line we ask for the X or Y coordinates of the $i\,th$ point. After calculating the area is printed on the console with a message.

0.4 Testing

0.4.1 Automated

We wrote a python script that generates random coordinates, runs the asm file, and compares the output with the expected area within a margin of error that can be specified while testing. We use spim command line tool to automate the testing using the python script. To run the script, we enter the following commands once inside root project folder

```
sudo apt install spim
python3 -m pip install -r requirements.txt
python3 tester.py -n 20 -m 100 -b 10 -e 0.0001
```

This will run 20 random test cases with 100 coordinates of at max 10 bits each, with an error tolerance of 0.0001 percent.
We tested with different values of error tolerance and bitsizes, and found that overflow occured at 16 bit inputs and with 15 bit inputs, we could achieve an accuracy of about 10^{-4} %.

0.4.2 Manual

We also checked a few corner cases manually in QtSpim, which are listed below :

1. number of points 0 or 1
2. all points on X-axis.
3. All points on Y-axis.
4. Duplicate points.

In all these cases we got the expected output without overflows, that is 0.