CSE 341: COMPUTER ORGANIZATION Project (FALL 2020)

Project Due: December 10, 2020

Project Description

In this project, you are required to design and implement the algorithm for *Closest Pair of Points Problem* in assembly language.

The *closest pair of points problem* is a problem of computational geometry: given n points in metric space, find a pair of points with the smallest distance between them. The closest pair problem for points in the Euclidean plane was among the first geometric problems that were treated at the origins of the systematic study of the computational complexity of geometric algorithms. You may refer to the Wikipedia for detailed explanation of the problem.

A *sample data* will be given to you for test purpose. For the evaluation process, TA will execute your codes with unseen data points.

Submission Approach & Grading Criteria

Submission Approach: Implement your logic in the given files (see Table 1 below). **Do not** modify any source code file name or given code. Your report should be saved as a **PDF**. To submit, put 5 source code files and 1 report in the folder named using your UB personal number. Compress the folder as *your_ub_personal_number.zip* and submit. **TA will not grade submission organized with different hierarchy.** Below is an example of files hierarchy (assuming UB Personal number is 50291111):

```
50291111.zip
        |50291111
                |euclidean_distance.s
                |sort.s
                |find.s
                |io.s
                |main.s
                |report.pdf
```

Grading Criteria: TA will grade *find.s*, *sort.s*, io.s, and *euclidean_distance.s* separately, *e.g.*, you will get 20% pts if your Euclidean distance sub-procedure works correctly. *This enables partial pts* when your program cannot give correct outputs. All the tests will be based on unseen data.

Table 1: Project Files

Given Files	Function	Need Submission	pts
euclidean_distance.s	Calculate Euclidean distance	Yes	20%
sort.s	Sort points by x and y value	Yes	10%+10%
find.s	Find closest pair (Divide and Conquer)	Yes	20%
io.s	Input & output	Yes	15%+5%
report.docx	Editable report	Yes, **as PDF**	20%
sample.bin	Sample data points for your local test	No	N/A
run_qtspim.cmd	Run qtspim and load project files	No	N/A
CSE341B_Project.pdf	Project introduction	No	N/A

Late Submission Policy: Submission after the due (December 10, 2020) will not be considered.

Helpful resources

We will use the divide and conquer algorithm described in http://people.csail.mit.edu/indyk/6.838-old/handouts/lec17.pdf

You may use this implementation https://www.geeksforgeeks.org/closest-pair-of-points-using-divide-and-conquer-algorithm/ as your reference to understand the divide and conquer algorithm.

Project Details

Project Framework (main.s)

The *main.s* file uses your implementation to find the closest pair of points of the given data points. You **should not** modify the code in this file. However, take a look at this file to understand 1) how the arguments are passed to different functions; 2) how the return values are used

Data Structure (sample.bin)

The sample data is given in the sample.bin file. This file has the following binary structure

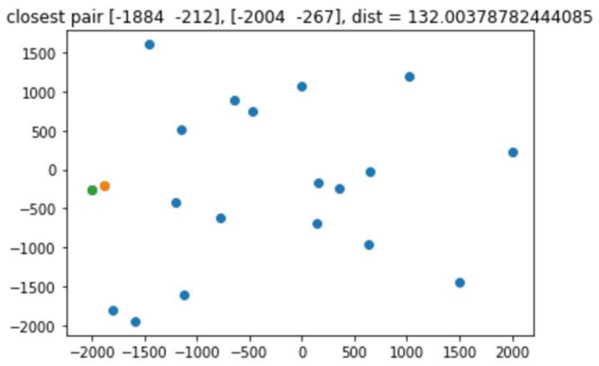
|32 bits (num_points)| 32 bits (x0)| 32 bits (y0)| 32 bits (x1)| 32 bits (y1)|…| 32 bits (xn)| 32 bits (yn)|

Where *num_points* indicates how many points there will be in the file. Also, num_points <= 2048, abs(*x*)<=2048, abs(*y*)<=2048. The binary is little-endian. During grading we will use the unseen data that conforms to the same restriction. The distribution of sample data is shown in the following figure

allocate num_points $\times \underset{\text{bytes}}{4} \times \underset{(x,y)}{2}$

all points <= 2048 $\rightarrow$ $2048^2 < 2^{31}$

mult → mflo
omit mfhi



Input and output (io.s)

In this file, you are supposed to implement three functions:

1. *load_points*
   Given a file name, this function uses syscall 13/16 to open and close the file. After opening the file, this function passes the file descriptor to the *load_points_helper*. The helper is responsible for returning a) the number of points in the file (*num_points*); b) the base address of the array containing all the point coordinates. When *load_points_helper* returns, this function closes the file and returns *num_points* and base address. *When open the given file, use mode 0 and flag 0.*
2. *load_points_helper*
   Given a file descriptor, load data in the file and return a) the number of points in the file (*num_points*); b) base address of the array containing all the point coordinates. You should use syscall 14 to read and syscall 9 to allocate a memory block for the array. When loading data into array, use the order as x0,y0,x1,y1,…,xn,yn. The file binary format is introduced in *Data Structure*
3. output_closest_pair
   This function outputs the coordinates of the closest pair of points to the console. You can use *str_prefix_output* defined at the beginning of the file
   the output should be a one-line in the console as following

   The closest pair of points is x0,y0,x1,y1,

Euclidean Distance (Euclidean_distance.s)

Euclidean distance defines the distance between two points in Euclidean space. When considering a two-dimension space, the distance between A and B is given by:

$$dist(A,B) = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$$

In this project, for the convenience of integer calculations through the entire flow, you need to calculate the "modified" Euclidean distances between every two data points. Basically, you don't need to calculate the square root.

$$\boxed{dist'(A,B) = (x_A - x_B)^2 + (y_A - y_B)^2}$$

Because

$$dist(A,B) > dist(C,D) === dist'(A,B) > dist'(C,D)$$

In *Euclidean_distance.s* file, you are supposed to implement the above simplified Euclidean distance calculation and return the result.

Sort (sort.s)

In this file, you are supposed to implement two functions, which sort (all or a part of) the points by x and y coordinates, respectively. The sort is in-place so only the base address of the points array is passed as an argument. No return value is needed.

Find closest pair of points (find.s)

In this file, you implement the find function to find the closest pair of the points. If you use brute-forth methods (no recursion) and the results are correct, you will receive half credits.

Project execution (run_qtspim.cmd)

To avoid load project files every time, we provide a cmd file to automatically run qtspim with project files loaded. Double click it to run. To use it, open the cmd file using any text editor and specify the path of your qtspim installation and project folder in the first two lines. If you don't use this script, the file name in *main.s* should be absolute path.

Function signature

You can find the convention of arguments and return values (signature) in the code files. They are written as comments.

Report

See report.docx for details