TEE2028 Microcontroller Programming and Interfacing

Lab Assignment 1

Group 3

Lecturer: Gu Jing

Group members:
Low Feng Rui (A0214782A)
Celine Oi Shu Jun (A0214749X)

Question 1: Knowing the starting address of array points10*[][]*, how to find the memory address of the A-th data point, say A ≤ M ? Drawing or equations can be used to explain your answer.

Knowing the starting address of the data points, to move to the memory address of the next data point we just have to offset by 4 bytes. For example if the starting address is stored in [R0] and we want to load the next data point into R1, we can offset [R0] by 4 bytes and load the value in that memory address into R1. Eg. LDR R1,[R0,#4]!. In the same way, we can access the next data point using LDR R1,[R0,#8]! since each data point takes up 4 bytes of memory space.

Below is an example of the offset required to access data points in an array.

[R0]	0.0	0.0	[R0,#4]
[R0,#8]	0.0	1.0	[R0,#12]
[R0,#16]	1.0	1.0	[R0,#20]
[R0,#24]	1.0	0.0	[R0,#28]
[R0,#32]	3.0	0.0	[R0,#36]
[R0,#40]	3.0	1.0	[R0,#44]
[R0,#48]	4.0	0.0	[R0,#52]
[R0,#56]	4.0	1.0	[R0,#60]

Question 2: Describe what you observe in (i) and (ii) and explain why there is a difference.

```
Class for each point:
point 0: class 0
point 1: class 0
point 2: class 0
point 3: class 0
point 4: class 0
point 5: class 0
point 6: class 0
point 7: class 0

New centroids:
(0.8, 0.2)
(0.0, 0.0)
```

I) Output values are printed out after running the program

II) There are no output values printed out.

When the program is branching from “main.c” to “classification.s”, the return address is being stored into {R14}. Hence, without the “push” and “pop”, there is no address for the program to return to “main.c”. R{14} is used to store the return program counter (PC) when a subroutine or function is called.

Question 3: What can you do if you have used up all the general purpose registers and you need to store some more values during processing?

We can either store the value in the general purpose register we want to use in memory using STR function or we can use PUSH and POP to temporarily store the value into the stack before retrieving it to use later.

For example, if we need to use R1 to store some more values during processing but it is already used up, we can use PUSH to store the current value into the stack temporarily. R1 can then be used to store other values for processing. Afterward, we just have to POP to “restore” R1 to its original value when we need it.

Program Description

Introduction

The program is to simulate K-means clustering through the means of unsupervised machine learning. The initial centroids used to identify subgroups in the data points may not be accurate. Hence the main point of the 2 sections in this program is to find out new centroids(characteristics) to better identify and classify the data points into subgroups.

There are 2 parts to this program, “classification.s” and “find_new_centroids.s”.

Classification.s

This part of the program is to group the data points into Class 1 or Class 2, which will be used to calculate the new centroids for splitting the data points into their new subgroups.

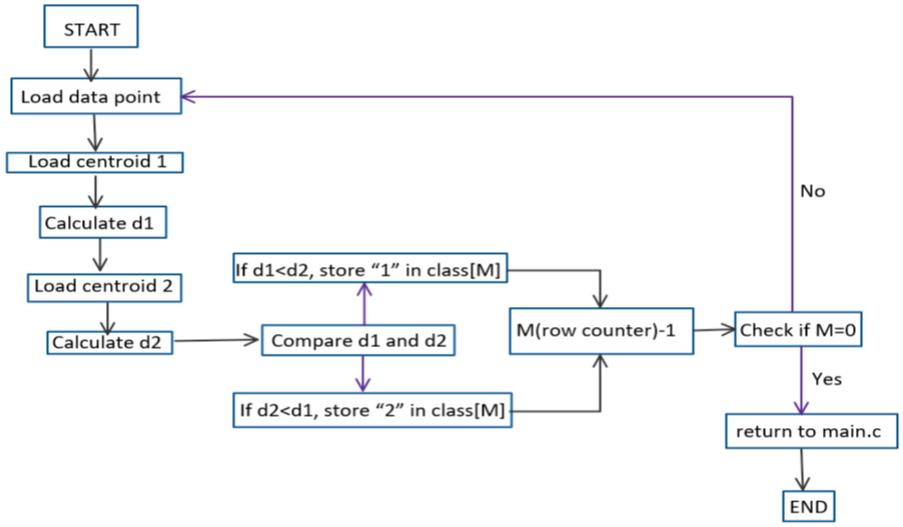
Steps for classifying into class 1 or class 2

1. Calculate distance(d1) between centroid 1[x_1,y_1] and data point[x_2,y_2] using below formula
2. Calculate distance(d2) between centroid 2[x_1,y_1] and data point[x_2,y_2] using below formula

$$d = (x_2 - x_1) + (y_2 - y_1)$$

3. Compare d1 and d2. If d1<d2, classify the data point as Class 1. If d2<d1, classify the data point as Class 2.

Program Flowchart



To exit the loop after all the data points have been calculated, a counter M is introduced in the program.

```
.equ M,8          @set no. of rows of data points to 8, can be changed.
```

M refers to the number of rows of data points. After the completion of each loop(calculation of the set of data points), the counter will decrement by 1 until it reaches 0, whereby all rows would have been calculated and stop the loop.

Find_new_centroid.s

This part of the program is to re-compute the new centroids using data points that have been classified into Class 1 and Class 2 in **classification.s**

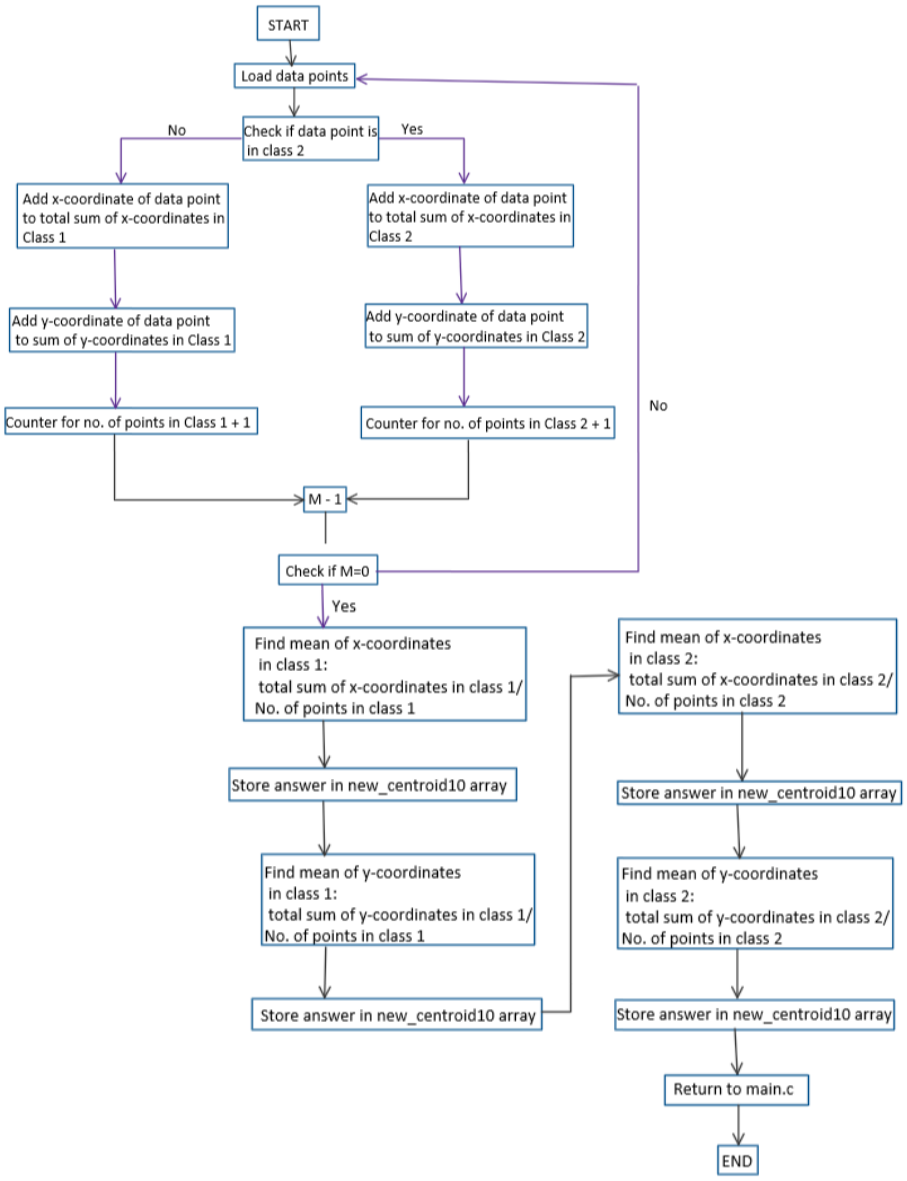
Steps to calculate the new centroids for each class

1. Find Mean of all x-coordinates of data points in Class
2. Find Mean of all y-coordinates of data points in Class

$$x_{centre} = \mu_x$$
$$y_{centre} = \mu_y$$

3. Update (x_{centre},y_{centre}) as new centroids for the class.

Program Flowchart



Improvements

The program was written in hardcode at first to simulate using only 8 rows of data. It was later rewritten with a counter M to make it more versatile, whereby M denotes the number of rows of data in the array so the program can adapt to different array sizes.

classification.s

Improvement 1

Originally, the coding in Find_dist was repeated 2 times for calculating d1 and d2 separately. As this is rather inefficient, repeated codes are taken out as a branch to make the code more simple.

```
BL Find_dist      @jump to Find_dist function to calculate
ADD R8,R4,R5      @R4+R5=d1, store d1 in R8 (d1=R8)
```

Improvement 2

As there were not enough general purpose registers to process calculations for d1 and d2 separately, PUSH and POP were also used to free up registers.

```
@Store current values of R4 and R5(data points) in stack since value will be changed in calculation
  PUSH {R4-R5}      @store data points in stack to retrieve later

@Retrieve data point (R4 and R5 before changes)
  POP {R4-R5}
```

find_new_centroid.s

Improvement 1

The original idea was to find the data points in class 1 and calculate centroid 1, and then repeat the same steps for centroid 2. However the process would take 2 separate loops, one for each class which makes the code unnecessarily lengthy. It was also more troublesome to access the memory locations in R0 as post-index offset has been used to calculate centroid 1 and we will have to “restore” R0 to the very first memory location in order to repeat the process again.

Improvement 2

The branch loop was added after the coordinates was calculated instead of after storing the values in the new centroid array, so that the total memory space needed for the program could be shortened. Hence saving the file size and cutting short the time needed for the program to be executed.