# PCI-2 [Week 4]: S1 AY2021-2022 Assignment 1

**********************************************************************************************

**Any further updates will be posted here...**

- **For BEng students: Submission folders are created in LumiNUS, please submit ONE report and ONE zipped workspace per group, following the requirements indicated in Section 4.**
- **The report should not exceed 4 pages, and you don't need to paste the entire version of the code into it.**
- **Please include the following in the report:**
  - **answers to the questions asked in the manual,**
  - **discussions of your program logic,**
  - **discussions of the improvement you have done or can be done in the future,**
  - **and whatever efforts you have done and you want to let the teaching team know.**
- **The centroid number N can be assumed as 2 all the time.**

**********************************************************************************************

## 1. Background Concepts

K-means clustering is one of the popular algorithms in unsupervised machine learning. It can be defined as the task of identifying subgroups in the data such that data points in the same subgroup (cluster) are very similar while data points in different clusters are very different.

Illustrated by Figure-1, given the initial centroids $C_1$ and $C_2$, a set of data points can be classified as class1 (red) or class2 (blue) based on the distances between the points to the respective centroids. For example, point P1 is closer to C1 as compared to C2, therefore it is classified as a class1 point. After the classification of all data points, new centroids will be computed by taking the average of all data points that belong to each class. K-means clustering usually takes several iterations of the procedure until there is no change to the centroids.

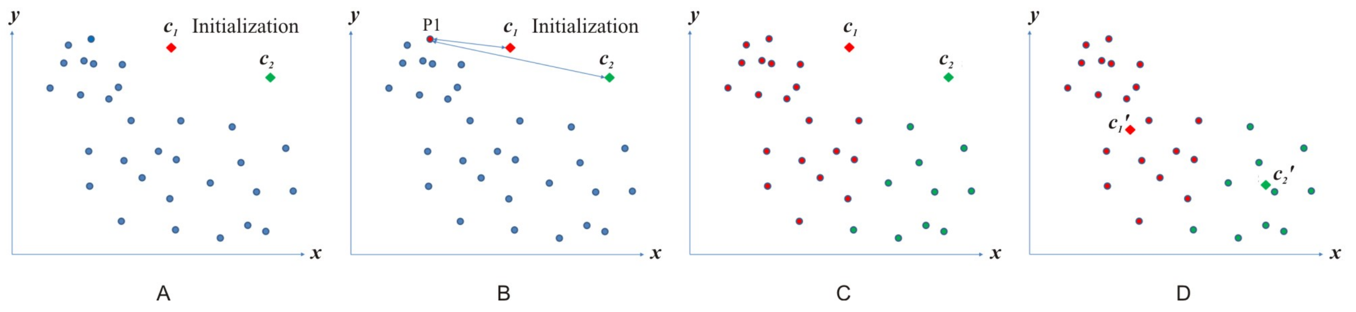


*Figure-1. K-Means Clustering of Binary Classification (binary classification refers to the classification of all data points into 2 groups).*

## 2. Objective

The objective of this assignment is to develop two ARMv7-M assembly language functions **classification()** and **find_new_centroids()** that classify the data points based on the initial centroids and then re-compute the new centroids.

## 3. Procedure

### (a) Preparation

Download Assignment 1 template which contains the "EE2028_AY2122S1_Assign1" project and unzip it into your EE2028workspace.

In STM32CubeIDE, choose ***File > Open Project from File System... > Directory... > browse to your workspace and choose the folder called "EE2028_AY2122S1_Assign1"***. Check "***Add project to working sets***". Click ***Finish***.

• **classification.s**, which presently contains only a few instructions – this is where you will write the assembly language instructions that implement the **classification()** function to classify the data points into 2 groups; and

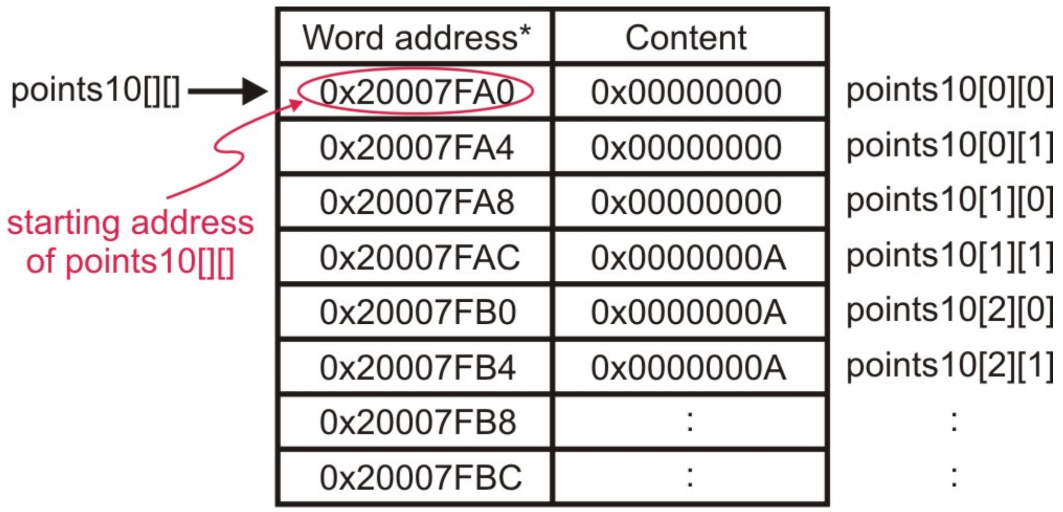
• **find_new_centroids.s**, which presently contains only a few instructions – this is where you will write the assembly language instructions that implement the **find_new_centroids()** function to re-compute the new centroids based on the classification result; and

• **main.c**, which is a C program (to be described below) that calls the **classification()** and **find_new_centroids()** function with the appropriate parameters and prints out the result on the console pane. You do NOT need to modify this file unless you want to test your assembly language function with different values in the arrays or a different number of data points M.

### (b) Initial Configuration of Programs

The C program **main.c** in the project "EE2028_AY2122S1S1_Assign1" defines a 2D array ***points[M][2]*** that contains the coordinates of **M** number of data points, and a 1D array ***centroids[N][2]***that defines the initial centroids. In order to do computation using functions written by assembly instructions, the coordinates are multiplied 10 times larger to pass into the functions. Therefore, the 2D array ***points10[M][2]*** and 1D array ***centroids10[N][2]*** are defined and initialized in **main.c**.

The elements of a 2D array are stored row by row in consecutive words in memory starting from word address that can be found by its pointer value, illustrated in Figure-2. For example, the starting address of ***points10[][]*** can be found by its pointer **(int*)points10**.



| Word address* | Content | |
|---|---|---|
| 0x20007FA0 | 0x00000000 | points10[0][0] |
| 0x20007FA4 | 0x00000000 | points10[0][1] |
| 0x20007FA8 | 0x00000000 | points10[1][0] |
| 0x20007FAC | 0x0000000A | points10[1][1] |
| 0x20007FB0 | 0x0000000A | points10[2][0] |
| 0x20007FB4 | 0x0000000A | points10[2][1] |
| 0x20007FB8 | : | : |
| 0x20007FBC | : | : |

*Figure-2. Illustration of contents of consecutive memory locations containing elements of the array points10[][].*

**Note: The actual memory addresses for each array in your case may be different from the figure shown here.*

The C program calls the function **classification()** written by assembly language to determine the class of all M data points and update the 1D array ***class[]***, and then calls another assembly function **find_new_centroids()** to re-compute the new centroids and stores their coordinates into the 2D array ***new_centroids10[][]***.

Note: The ARMv7-M assembly language functions **classification()** and **find_new_centroids()** that you develop should be able to perform the required computation for any reasonable value of **M** (total number of data points).

**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 equation can be used to explain your answer.**

*Hint: if A-th data point is located from points10[Ax][Ay], formulate the relation between the memory address and the indices.*

In the C program **main.c**, the function definition for **classification()** and**find_new_centroids()** is **extern void classification/find_new_centroids (int* arg1, int* arg2, int* arg3, int* arg4)**, which accept the following four parameters:

| Arguments | Descriptions |
|---|---|
| *points10[M][2]* | (10 times of x and y coordinates of M number of data points) |
| *cenroids10[N][2]* | (10 times of x and y coordinates of N number of initial centroids, typically N=2) |
| *class[M]* | (an empty array that to be used to store the class of each data point) |
| *new_centroids10 [N][2]* | (an array with parameters **M** and **N** passed as the 1st and 2nd elements at the start. This array is to store the 10 times of x and y coordinates of the new centroids) |

### Parameter passing between C program and assembly language function

In general, parameters can be passed between a C program and an assembly language function through the ARM Cortex-M4 registers. In the function **extern int asm_func (arg1, arg2, ….)**:

**arg1** will be passed to the assembly language function asm_func() in the **R0** register, **arg2** will be passed in the **R1** register, and so on. Totally 4 parameters can be passed from the C program to the assembly program in this way.

If there is a **return value** from the assembly language function back to the C program, the **R0** register should be used to keep the return value right before the program comes back to **main.c**.

In the function written by assembly language, there is a subroutine called **SUBROUTINE** declared after the main part of the assembly program, in order to demonstrate the way to "create and call a function" in an assembly language program. Note: The purpose of this declaration is to lead to the thinking of link register (LR/R14). At the completion of this assignment, **SUBROUTINE** is not compulsory to be used, i.e. you may not have BL SUBROUTINE in your function **asm_fun()**.

### (c) Warm Up

(i) Compile the "EE2028_AY2122S1S1_Assign1" project and execute the program.

(ii) Comment the **PUSH {R14}** and **POP {R14}** lines in **classification()**, recompile and execute the program again.

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

### (d) Overview of this Assignment

Write the code for the assembly language functions **classification()** and **find_new_centroids()** to fulfill the objective mentioned in Section 2 after reading the following aspects carefully.

• It is a good practice to push the contents in the affected general purpose registers onto the stack prior to or upon entry into the assembly language function or subroutine and to pop those values at the end of the assembly language function or subroutine to recover them.

• If you are using a subroutine, special care must be taken with the Link Register (R14). If the content of this register is lost, the program will not be able to return correctly from the subroutine to the calling part of the program.

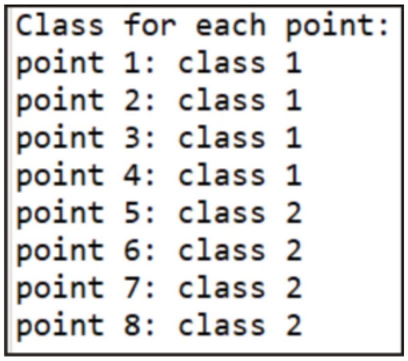
• If a set of actions are conditional, you may use the ARM assembly language IF-THEN (IT) block feature or conditional branch.

• In a RISC processor such as the ARM, arithmetic and logical operations only operate on values in registers. However, there are only a limited number of general purpose registers, and programs, e.g. complex mathematical functions, may have many terms.

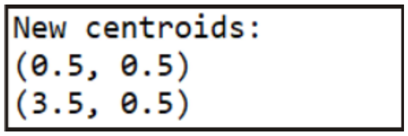
*Hint: Use and re-use the registers in a systematic way. Maintain a data dictionary or table to help you keep track of the storage of different terms in different registers at different times.*

**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?**

Verify the correctness of the results computed by the **classification()** and **find_new_centroids()** functions you have written by comparing what appears at the console window of the STM32CubeIDE with the desired output shown below:



```
Class for each point:
point 1: class 1
point 2: class 1
point 3: class 1
point 4: class 1
point 5: class 2
point 6: class 2
point 7: class 2
point 8: class 2
```



```
New centroids:
(0.5, 0.5)
(3.5, 0.5)
```

During the assessment session in week 6, show the assembly language program and actual console output to the Graduate Assistant (GA). You are also required to write a short report that answers all the questions asked in the manual and elsewhere in this manual, and describes how you have implemented the assembly language functions. See Section 4 below on the assessment session.

## 4. Assignment Submission (Week 6)

The Assignment 1 LumiNUS submission deadline is 1 hour before the assessment, as shown below. There will be 1-hour grace time, however, try NOT do last-minute submission to prevent traffic congestion.

| Lab Class | Assessment 1 (week 6 lab time) | Deadline of LumiNUS submissions |
|---|---|---|
| EE2028-B01 (Tuesday AM Session) | 9AM-12PM, 14-Sep | 8AM, 14-Sep |
| EE2028-B02 (Tuesday PM Session) | 2PM-5PM, 14-Sep | 1PM, 14-Sep |

| | | |
|---|---|---|
| EE2028-B03 (Friday PM Session) | 2PM-5PM, 17-Sep | 1PM, 17-Sep |
| TEE2028-B01 (BTech, Tuesday night Session) | 6PM-9PM, 14-Sep | 5PM, 14-Sep |
| TEE2028-B02 (BTech, Wednesday night Session) | 6PM-9PM, 15-Sep | 5PM, 15-Sep |

### Please follow the rules here to name your archived project and report (ONE set of submission / group):

Get your group number from LumiNUS, and name the project archive file and report (must be in PDF format) that to be submitted (DO NOT zip them together) as following naming templates:

**• For the Project (only ZIP file is accepted):**

Assignment1_Group Number_ Matriculation number of all team members separated by an underscore_Workspace.zip

Example: Assignment1_Tue-AM01_A0131086Z_A0010101Z_Workspace.zip

**• For the report (only PDF file is accepted):**

Assignment1_Group Number_ Matriculation number of all team members separated by an underscore_Report.pdf

Example: Assignment1_Tue-AM01_A0131086Z_A0010101Z_Report.pdf

### Please do the following steps to prepare your Assignment 1 project archive and report that to be submitted to LumiNUS:

(1) In your **classification.s** and **find_new_centroids.s** files, please include the names and matriculation numbers of your team members (As comments on the first few lines)

(2) In the STM32CubeIDE software, right-click the project (**DO NOT** include any example projects i.e. blink programs from lab1 into the archive file) from **Project Explorer** > **Export** > **Archive File** > ***check ONLY the assignment project and all its components to be exported*** > ***choose the location to export and name the archive file as described above***.

(3) Make sure "**Create directory structure for files**" is checked, then click **Finish**.

## 5. Assessment and Contribution Declaration (Week 6)

The assessment schedule will be published soon. Please present in the lab **at least 15 minutes** before your personal time slot.

Note: You will not be allowed to use your own laptop during the assessment. Therefore, please ensure that your program has been tested on the desktop PC in the lab.

When your group is called by the GA, bring along the (a) **STM32 board**, (b) **hard-copy report** and (c) **a personal thumb drive containing your archived project** to the assessment station.

During the assessment, show the execution of your program and the console output from the program and explain how your program works. The GA will modify your program slightly, e.g. use arrays with different values and sizes M to assess whether the correct outputs can be determined.

The assessing GA will look at both aspects and will ask each student some questions in turn during the assessment. Therefore, **both students need to be familiar with all aspects of the assignment and your solution** since the GA can choose any one of you to explain any part of the assignment, i.e. you need to know what your partner has done. The GA will also ask you additional questions to test your understanding.

**Note: Each team member should make both joint and specific individual contributions towards the end results of this assignment.**

After the assessment, each student needs to complete the online declaration of contribution through **LumiNUS** > **[T]EE2028** > **Survey** > **Contribution Declaration (Assignment 1)**. The deadline for online declaration is **12PM (noon) 19-Sep (Week 6 Sunday)**. The results from the online Contribution Declaration will be cross-checked with inputs from the assessing GA to moderate the final individual's mark. Your inputs will be treated as confidential - your partner will not be able to see how you rated them (so you can and should be honest - the marks released will not reflect the Contribution Declaration inputs though, which will be incorporated later).

**- THE END -**