CG2028 Project Report:

K-nearest neighbours (k-NN) in assembly

# Project Members:

|  |  |
| --- | --- |
| Nigel Ng | A0217408H |
| Mayank Panjiyara | A0221571N |

# Purpose:

This report aims to explain how we implemented classification.s, as well as the machine code and modified architecture to support MLA, MUL.

# Testing:

The following code represents the behaviour of our program with different inputs, where the print statements illustrate the:

1. result of classification.s
2. calculations of Euclidean distance in c
3. output in c matching (1.)

Table

Description automatically generatedText

Description automatically generated with low confidenceTable

Description automatically generated

# Key assumptions on implementation:

Following the project instructions, we assumed the following

1. Inputs from main.c are validated and instantiated accordingly.
2. Euclidean distances are guaranteed to be distinct for different points
3. Labels for points are either 0 or 1.
4. There would never be any overflow during calculations; any values used at any time in classification.s are guaranteed to be within 32 bits
5. Sample x and y coordinates are stored in successive addresses in points[]. (x1, y1, x2, y2 …)
6. K is 1.

# Classification.s flowchart

The flowchart provides a high-level overview of classification.s.

Diagram

Description automatically generated

# Code

Register map

@R0 - N, returns class

@R1 - pointer to points

@R2 - pointer to label

@R3 - pointer to sample

@R4 - stores point

@R5 - stores sample

@R6 - (point[i] - sample[i]), (point[i] - sample[i])^2, label[i]

@R7 - (point[i+1] - sample[i+1]), (point[i+1] - sample[i+1])^2, CURRENT\_DISTANCE

@R8 - CURRENT\_DISTANCE\_LABEL

@R9 - MIN\_DISTANCE

@R10 - MIN\_DISTANCE\_LABEL,class

classification:

    PUSH {R1-R10,R14}

   BL EUCLIDEAN\_DISTANCE

   LDR R10, [R2], #4 @0b0000 0101 1001 1010 0010 0000 0000 0100 0x059A2004

   MOV R9, R7 @0b0000 0001 1010 0111 1001 0000 0000 0000 0x01A79000

SUBS R0, #1 @0b0000 0010 0101 0000 0000 0000 0000 0001 0x02500001

 LOOP:

BL EUCLIDEAN\_DISTANCE

LDR R8, [R2], #4 @0b0000 0100 1001 0010 1000 0000 0000 0100 0x04928004

CMP R9, R7 @0b0000 0001 0101 1001 1001 0000 0000 0000 0x01599000

ITT PL

MOVPL R9, R7 @0b0000 0001 1010 0111 1001 0000 0000 0000 0x01A79000

MOVPL R10, R8

SUBS R0, #1 @0b0000 0010 0101 0000 0000 0000 0000 0001 0x02500001

BNE LOOP @0b0001 1000 0000 0000 0000 0000 0010 0000 0x18000020

 MOV R0, R10 @0b0000 0001 1010 0000 0000 0000 0000 1010 0x01A0000A

    POP {R1-R10,R14}

    BX LR

EUCLIDEAN\_DISTANCE:

LDR R4, [R1], #4 @0b0000 0100 1001 0001 0100 0000 0000 0100 0x04914004

LDR R5, [R3] @0b0000 0101 1001 0011 0101 0000 0000 0000 0x05935000

SUB R6, R4, R5 @0b0000 0000 0100 0100 0110 0000 0000 0101 0x00446005

MUL R6, R6 @0b0000 0000 0000 0000 0110 0110 0001 0110 0x00006616

LDR R4, [R1], #4 @0b0000 0100 1001 0001 0100 0000 0000 0100 0x04914004

LDR R5, [R3, #4] @0b0000 0101 1001 0011 0101 0000 0000 0100 0x05935004

SUB R7, R4, R5 @0b0000 0000 0100 0100 0111 0000 0000 0101 0x00447005

MLA R7, R7, R7, R6 @0b0000 0000 0010 0110 0111 0111 0001 0111 0x00247717

   BX LR

# Explanation

Colour scheme for machine code (just to identify substring)

Cond, Op, X, U, imm8, Rn, Rd Rm, L, U, W, P, I, Cmd, S, M

We used 10 general-purpose registers to handle data processing.

R0 to R3 are used for the input from main.c

R4 and R5 are used as intermediate registers to load the x/ y values

R6 and R7 are used for calculating the Euclidean distance for each point and stores it to CURRENT\_DISTANCE

R8 is used to get the label for the current point: CURRENT\_DISTANCE\_LABEL

R9 is used to store the MIN\_DISTANCE, to be compared with CURRENT\_DISTANCE

R10 is used to store the label for MIN\_DISTANCE : MIN\_DISTANCE LABEL, which returns the value to R0 when all points are iterated through.

Let an iterator of our point[] and label[] be i, where i = 0 initially

Firstly,

1. Calculate the current distance of the point[0], relative to sample, by passing both point and sample data into EUCLIDEAN\_DISTANCE (ED) subroutine.
2. Initialise MIN\_DISTANCE to the distance calculated by ED.
3. N = N - 1, i = i + 1, ‘SUBS’ to check the flags for our loop comparison later.

Next, we loop the remaining points

1. Pass point[i] into ED
2. Get label of point[i], CURRENT\_DISTANCE\_LABEL, save in R8
3. If CURRENT\_DISTANCE <= MIN\_DISTANCE
   1. MIN\_DISTANCE = CURRENT\_DISTANCE
   2. MIN\_DISTANCE\_LABEL = CURRENT\_DISTANCE\_LABEL
4. N = N - 1.
5. (N == 0)? break loop : i = i + 1

Finally, at this point due to the loop invariant (N != 0) being broken, all points are iterated through, and the MIN\_DISTANCE\_LABEL represents the closest class the sample identifies with N neighbours.

In this subroutine we achieve the following:

1. Load value of sample[0], sample[1] corresponding to x, y coordinates of sample
2. Load value of point[i], point[i + 1] corresponding to x, y coordinates of a point
3. Get
4. Store difference as CURRENT\_DISTANCE IN Register R7
5. Return to next line in LOOP

# Modified architecture to support MLA, MUL instructions