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.)

```
asm: class = 1
d1 = 1033, class = 1
d2 = 9593, class = 1
d3 = 21073, class = 0
d4 = 10090, class = 0
d5 = 8212, class = 1
d6 = 11840, class = 0
d7 = 6605, class = 1
d8 = 5224, class = 0
d9 = 11525, class = 1
d10 = 6850, class = 1
d11 = 8345, class = 0
d12 = 3445, class = 0
d13 = 1181, class = 1
d14 = 5378, class = 0
d15 = 7253, class = 1
d16 = 7865, class = 0
C  : class = 1
```

```
asm: class = 1
d1 = 700468, class = 1
d2 = 36818, class = 1
d3 = 627337, class = 0
d4 = 1460833, class = 1
d5 = 727376, class = 1
d6 = 1321658, class = 0
d7 = 966257, class = 1
d8 = 1547377, class = 1
d9 = 849433, class = 0
d10 = 615421, class = 1
d11 = 1594000, class = 1
d12 = 521053, class = 1
d13 = 318865, class = 1
d14 = 849074, class = 1
d15 = 915433, class = 0
d16 = 276778, class = 0
C  : class = 1
```

```
asm: class = 0
d1 = 87634781, class = 1
d2 = 130016594, class = 1
d3 = 114928354, class = 0
d4 = 95904185, class = 1
d5 = 72568162, class = 1
d6 = 63034148, class = 0
d7 = 100305284, class = 1
d8 = 32905841, class = 1
d9 = 9218570, class = 0
d10 = 24365249, class = 1
d11 = 72588073, class = 1
d12 = 95919385, class = 1
d13 = 30329737, class = 1
d14 = 46308114, class = 1
d15 = 25328554, class = 0
d16 = 4448336, class = 0
C  : class = 0
```

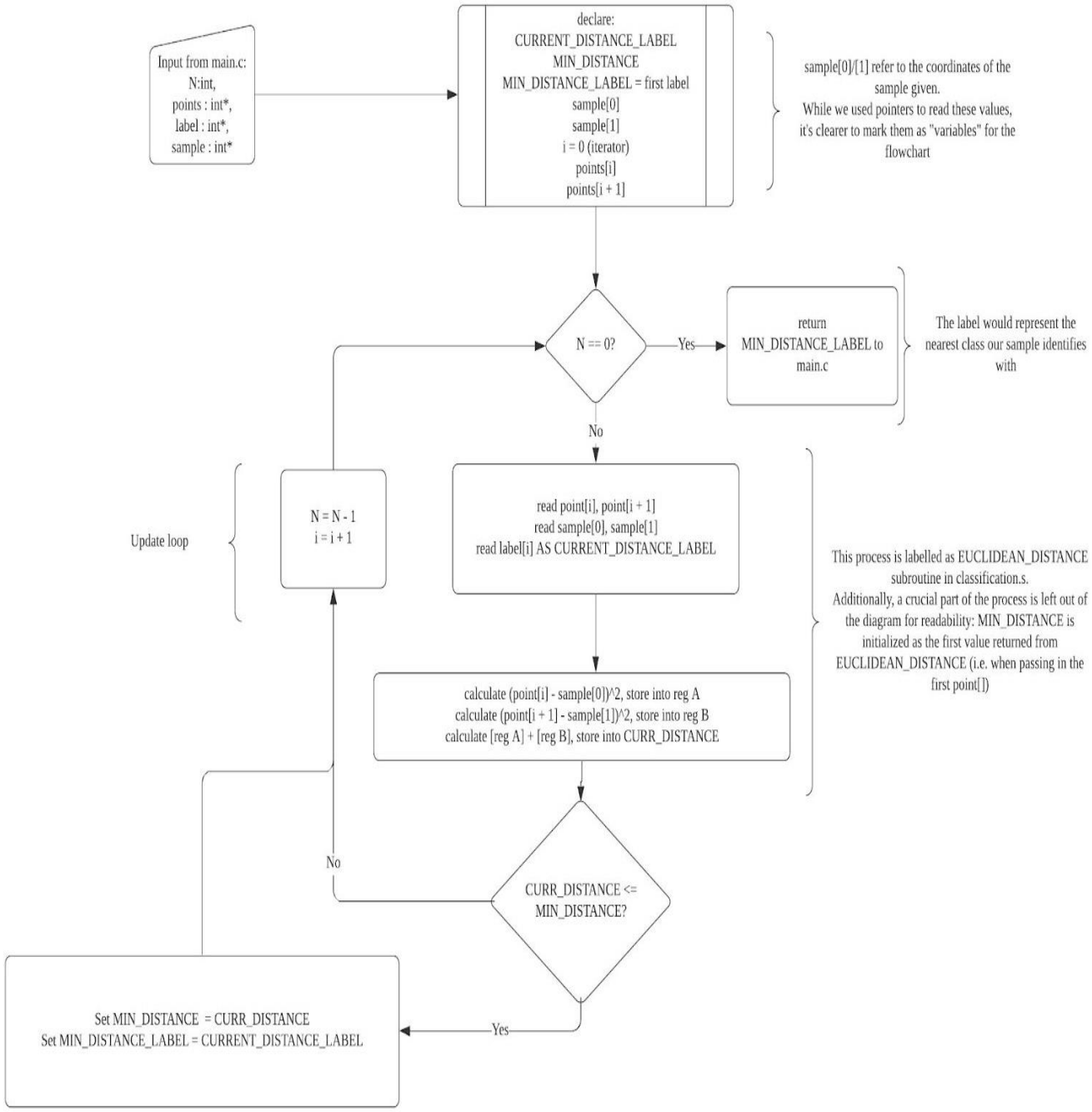
Key assumptions on implementation:

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[]. (x_1, y_1, x_2, y_2 …)
6. K is 1.

Classification.s flowchart

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



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 - (difference in x), (difference in x)^2
@R7 - (difference in y), (difference in y)^2, stores value of CURRENT_DISTANCE
@R8 – stores CURRENT_DISTANCE_LABEL
@R9 – stores MIN_DISTANCE
@R10 – stores MIN_DISTANCE_LABEL, class


classification:
    PUSH {R1-R10,R14}
          BL EUCLIDEAN_DISTANCE
          LDR R10, [R2], #4              @0b0000 0100 1001 0010 1010 0000 0000 0100 0x0492A004
          MOV R9, R7                     @0b0000 0001 1010 0000 1001 0000 0000 0111 0x01A09007
          SUBS R0, #1                    @0b0000 0010 1001 0000 0000 0000 0000 0001 0x02900001


 LOOP:
          BL EUCLIDEAN_DISTANCE
          LDR R8, [R2], #4               @0b0000 0100 1001 0010 1000 0000 0000 0100 0x04928004
          CMP R9, R7                     @0b0000 0001 0101 1001 0000 0000 0000 0111 0x01590007
          ITT PL
          MOVPL R9, R7
          MOVPL R10, R8
          SUBS R0, #1                    @0b0000 0010 1001 0000 0000 0000 0000 0001 0x02900001
          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 0x00267717
          BX LR
```

Explanation

Colour scheme for machine code (just to identify substring)
Cond, Op, X, U, imm8, R_n, R_d R_m, 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 $CURRENT\_DISTANCE = (Sample\_x - point[i]\_x)^2 + (Sample\_y - point[i]\_y)^2$
4. Store difference as CURRENT_DISTANCE IN Register R7
5. Return to next line in LOOP

Modified architecture to support MLA, MUL instructions

The following architecture (from Lecture 4 Slide 28) is modified to support MUL and MLA instructions.

The hardware multiplier block is represented by the symbol MUL (as it is used to perform MUL instruction).

Control signals MultControl and M are added and the ALUControl signal is modified.
The decoder also takes in another 1 bit input M to indicate if Multiplication operation is being performed.

The register file is also modified to take one more input A4 and allow reading it (RD4) simultaneously with RD1 and RD2.

```
ALUControl = (op==00)?(I==0 && M==1 ? 0100:cmd):((U) ? 0100:0010);
```

ALUControl now gives 0100 signal when MLA is specified to trigger the ALU block. I==0 and M==1 makes sure the ALU is not triggered while using an immediate in other DP operations.

```
MultControl = (op==00)&&(M==1)&&(I==0)&&(funt[1]==0);
```

MultConrol triggers the result from ALU to MUL when MUL instruction is performed.
Below are data paths for MUL and MLA respectively :