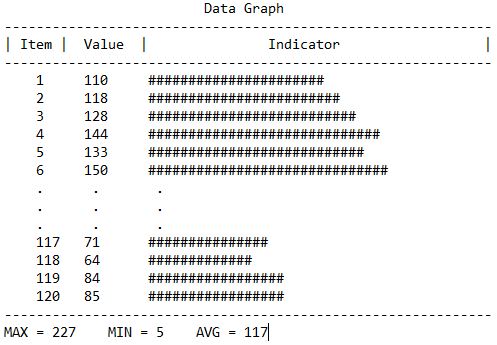
**Assignment 3: Mips mini project 2016**

**Author: Miroslaw Debiec / B00540796**

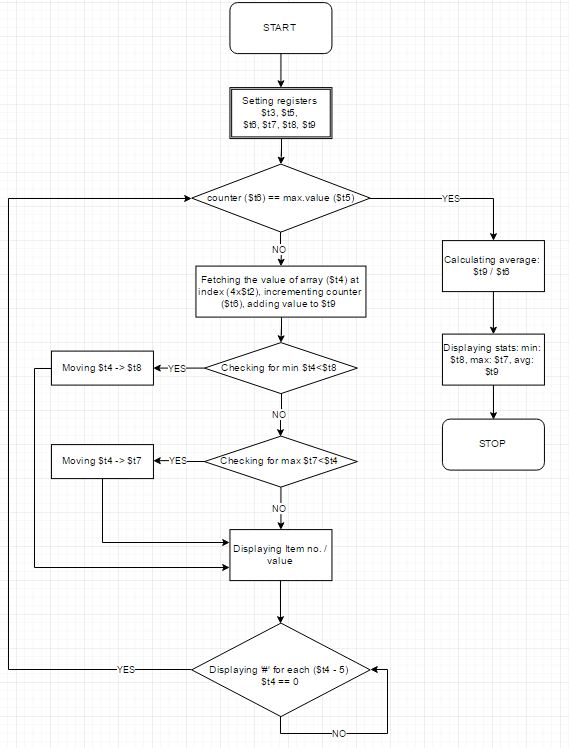
1. **Overview**

The file ‘CW3\_Debiec.s’ contains code in assembly language for simple graphical data representation of predefined integer values (Figure1.) The program outputs a table with 3 separate columns: first represents the item number which is also the index of the item in the array, second columns show actual value hold in the array in decimal format and the third column shows its graphical representation with ‘#’ signs.

At the bottom of the table, the program outputs simple data analysis with maximal, minimal and average values.

Figure1: Program output.

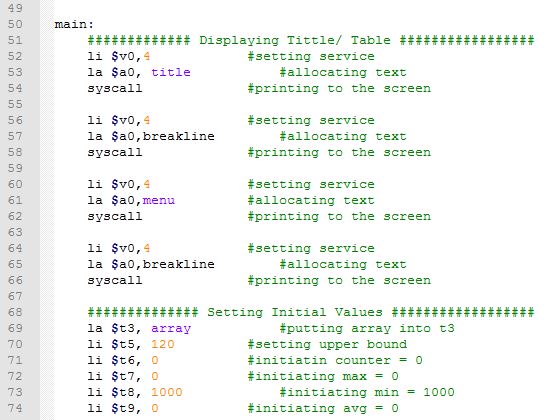
1. **Algorithm**

****Figure2. Shows the algorithm chart of the program. It is divided into several parts with outer loop for cycling through the array and performing the logic and arithmetic / logic operations on the given values and smaller inner loop for displaying and generating the representation of the held values at given index.

Once the main loop condition, program exits.

Figure2: Program algorithm

1. **Data Initialization:**

****Figure3. shows the initial steps performed by the program;

Lines 55-66 print the table form for easier reading of the data

Lines 69-74 setting the values to registers t3, t5-t9. Other t-registers are used to store temporary values throughout the code.

Figure3. Initialized data

1. **Loop: condition counter == maximum value (array length)**

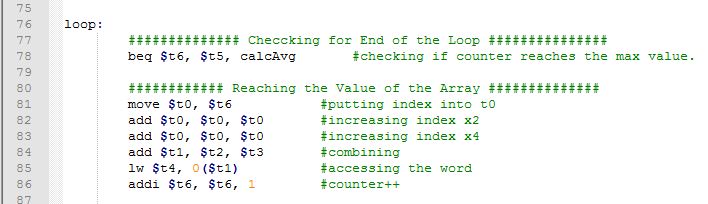


Figure4. Condition and array fetching

In Figure4. Line 78 checks for the counter value ($t6) equal to the maximum length ($t5), if the condition is met, it will jump into calculating the average value and terminating the program.

Line81, moves the value of counter into $t0 and increases the value by 4 (82-83), adds the index and address values in order to receive the value at given word (84 -85) and finally increments value of counter by 1.

1. **Checking for Minimum and Maximum values**

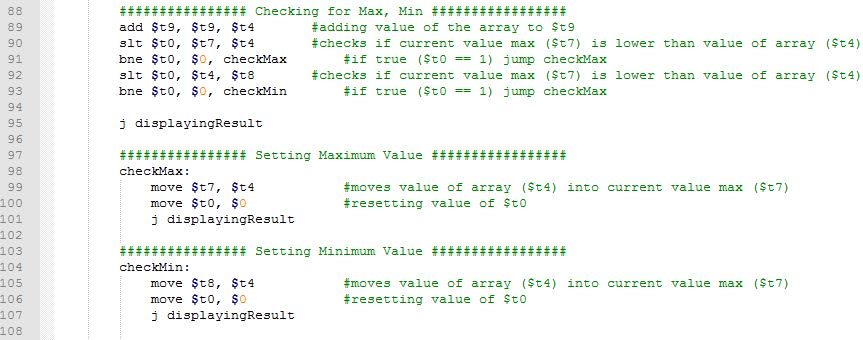


Figure5. Checking for minimum and maximum values

Line 88, adds the current value of the array $t4 to value in register $t9 and stores it in $t9 for another cycle.

Lines 90 – 91, checks if the current max value $t7 is lower than value of the array $t4 and if true, sets register $t0 to 1 that triggers jump to the ‘checkMax’ where it is being replaced with current highest value of $t4 and progresses with displaying line details (98 -101).

Similarly, lines 92-93, check if the current array value $t4 is lower than current stored min value $t8 and sets register $t0 to 1 that triggers jump to the ‘checkMin’ where it is being replaced with current lowest value of $t4 and progresses with displaying line details (104 -107).

1. **Displaying data: inner loop / graphical representation**

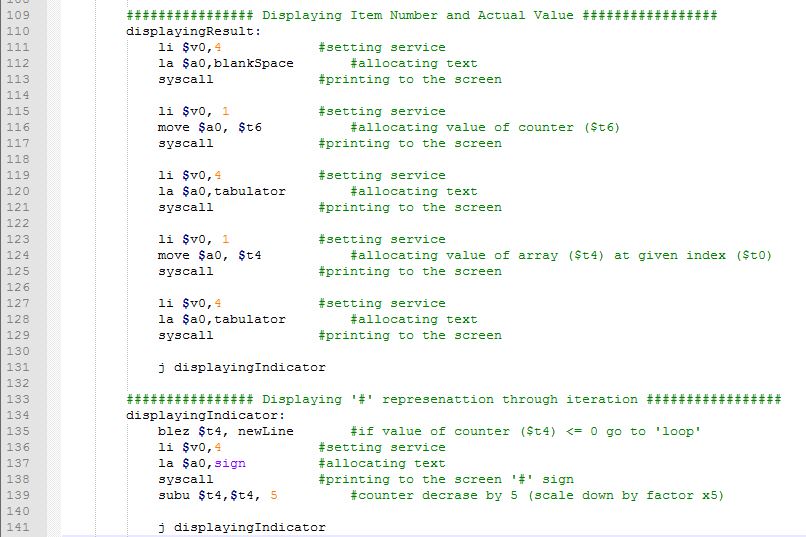
****

Figure6. Displaying new line data with ‘#’ graphical representation

Lines 110-129 display new line with counter number equal to item number of the array and its value, separated by predefined asci blank space and tab for better aesthetics of displayed data.

Line 131 triggers jump to inner loop (lines 134 – 141) where the value of $t4 is used as a counter and gradually decremented by 5 (139) displaying single ‘#’ character for each cycle until $t4 reaches value <= 0. If the final condition is met, the program jumps to newline part where it ends the current line in the table and returns to ‘loop’ (Figure4.).

1. **Calculating average**

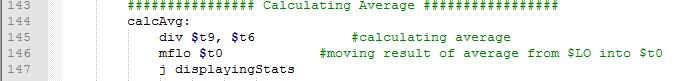


Figure 7. Calculating average.

Once the counter value $t6 reaches the maximum length of array ($t5), the loop jumps into calculating the average value. In order to do so it uses the value of the register $t9 that has been adding the values of each word in array ($t4) throughout the cycling and divides it by the number of cycles/indexes of the array that are stored in $t6.

This produces integer result in $LO register that is being recovered and moved into $t0 (line 146) and jumps to final section of the code where all the analytical results are displayed.

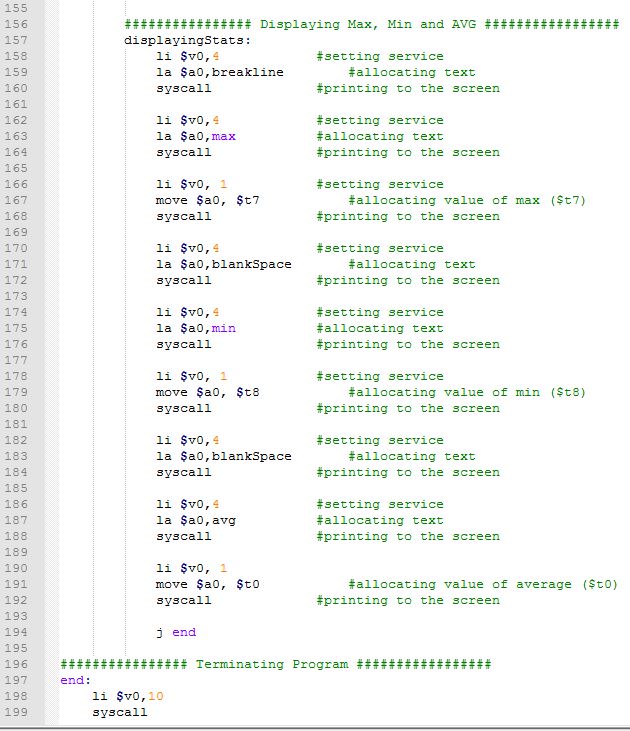
1. **Displaying data: analysis**

Figure8. Displaying results and ending the program

Lines 157 – 192 display the lower part of table (Figure1) with the maximal ($t7), minimal ($t8) and average ($t0) values separated by blank space for better aesthetics.

Once all data are displayed, the program jump finally to ‘end’ (lines 197- 199) where the assembler receives the instruction to terminate the program.

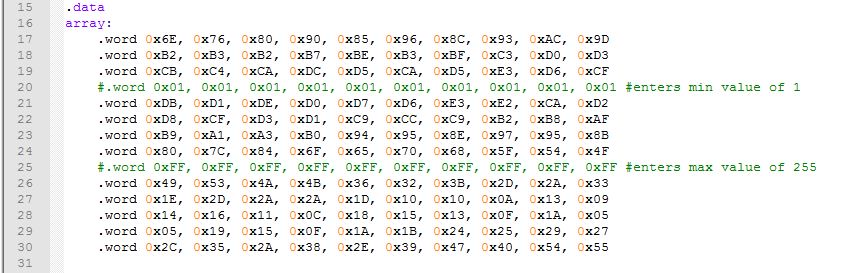
1. **Exercises**

Figure9.: Data section

Lines 17 – 30 represent the values stored in the array, and after running the program produce the result from Figure1 (MAX = 227 MIN = 5 AVG = 117). In order to test the code, the above values should vary when different data is being entered into the array.

There are two scenarios that can be used:

Scenario A:

Lower the values by uncommenting (removing ‘#’) from line 20 and using a comment on line 21 (Figure9a), this will set the values of items 31 – 40 in the array to 1 and alters the result (Figure10a). As we can see from the Figure10a, the results have changed (MAX = 227 MIN = 1 AVG = 99).

Scenario B:

Increase the values by uncommenting (removing ‘#’) from line 25 and using a comment on line 26 (Figure9b), this will set the values of items 71 – 80 in the array to 255 and alters the result (Figure10b). As we can see from the Figure10b, the results have changed (MAX = 255 MIN = 5 AVG = 133).

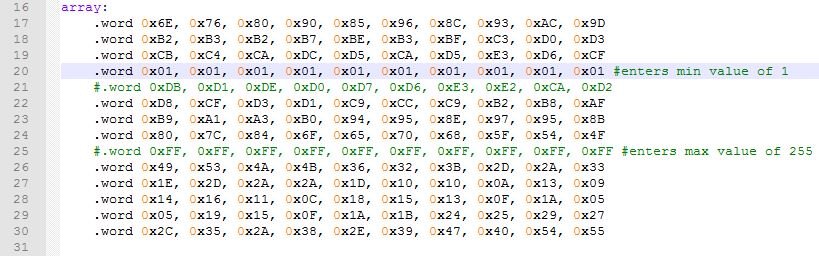
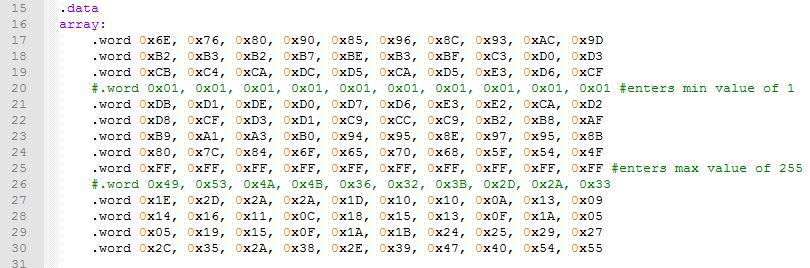
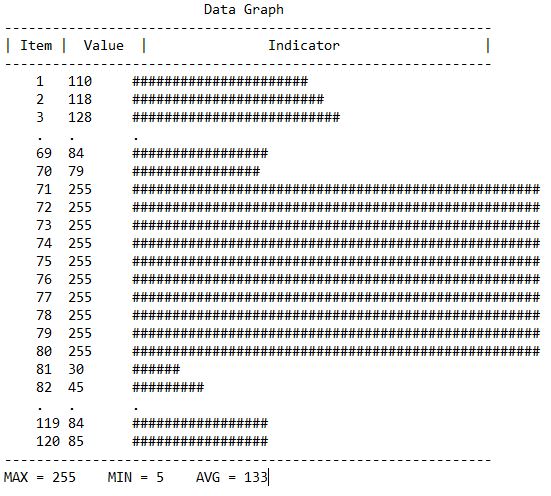
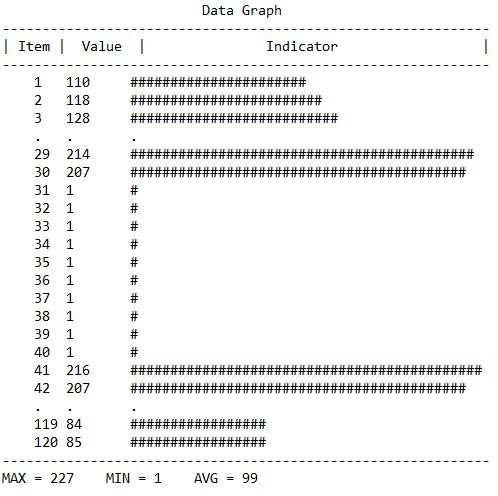
Figure9a:

Figure9b:

Figure10a: Figure10b: