## Assignment 3: Mips mini project 2016

**Author: Miroslaw Debiec / B00540796**

### 1. Overview

```
                    Data Graph
------------------------------------------------------------
| Item |  Value  |              Indicator                   |
------------------------------------------------------------
    1     110     ####################
    2     118     ######################
    3     128     ########################
    4     144     ############################
    5     133     ########################
    6     150     ############################
    .      .       .
    .      .       .
    .      .       .
   117    71      ##############
   118    64      ############
   119    84      ################
   120    85      ################
------------------------------------------------------------
MAX = 227    MIN = 5    AVG = 117
```

Figure1: Program output.

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.

### 2. Algorithm

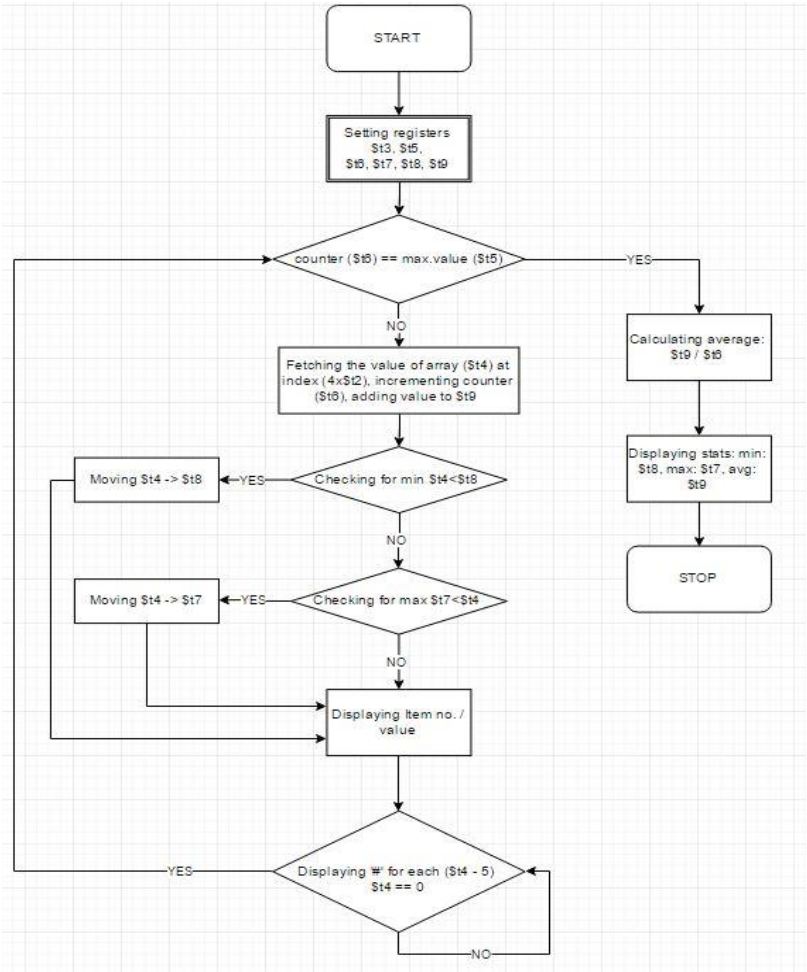


Figure2. Shows the algorithm chart of the program. It is divided into several parts with outer loop for cycling through the array and performing the logic and arithmetic / logic operations on the given values and smaller inner loop for displaying and generating the representation of the held values at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

Once the main loop condition, program exits.

Figure2: Program algorithm

### 3. Data Initialization:

```
49
50  main:
51      ############# Displaying Tittle/ Table #################
52      li $v0,4            #setting service
53      la $a0, title           #allocating text
54      syscall             #printing to the screen
55
56      li $v0,4            #setting service
57      la $a0,breakline        #allocating text
58      syscall             #printing to the screen
59
60      li $v0,4            #setting service
61      la $a0,menu         #allocating text
62      syscall             #printing to the screen
63
64      li $v0,4            #setting service
65      la $a0,breakline        #allocating text
66      syscall             #printing to the screen
67
68      ############## Setting Initial Values ##################
69      la $t3, array           #putting array into t3
70      li $t5, 120         #setting upper bound
71      li $t6, 0           #initiatin counter = 0
72      li $t7, 0           #initiating max = 0
73      li $t8, 1000            #initiating min = 1000
74      li $t9, 0           #initiating avg = 0
```

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

### 4. Loop: condition counter == maximum value (array length)

```
75
76  loop:
77          ############## Checcking for End of the Loop ###############
78          beq $t6, $t5, calcAvg       #checking if counter reaches the max value.
79
80          ############ Reaching the Value of the Array ##############
81          move $t0, $t6           #putting index into t0
82          add $t0, $t0, $t0       #increasing index x2
83          add $t0, $t0, $t0       #increasing index x4
84          add $t1, $t2, $t3       #combining
85          lw $t4, 0($t1)          #accessing the word
86          addi $t6, $t6, 1        #counter++
87
```

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.

## 5. Checking for Minimum and Maximum values

```
88          ################ Checking for Max, Min #################
89          add $t9, $t9, $t4       #adding value of the array to $t9
90          slt $t0, $t7, $t4       #checks if current value max ($t7) is lower than value of array ($t4)
91          bne $t0, $0, checkMax       #if true ($t0 == 1) jump checkMax
92          slt $t0, $t4, $t8       #checks if current value max ($t7) is lower than value of array ($t4)
93          bne $t0, $0, checkMin       #if true ($t0 == 1) jump checkMax
94
95          j displayingResult
96
97          ################ Setting Maximum Value #################
98          checkMax:
99              move $t7, $t4           #moves value of array ($t4) into current value max ($t7)
100             move $t0, $0            #resetting value of $t0
101             j displayingResult
102
103         ################ Setting Minimum Value #################
104         checkMin:
105             move $t8, $t4           #moves value of array ($t4) into current value max ($t7)
106             move $t0, $0            #resetting value of $t0
107             j displayingResult
108
```

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).

## 6. Displaying data: inner loop / graphical representation

```
109         ################ Displaying Item Number and Actual Value #################
110         displayingResult:
111             li $v0,4            #setting service
112             la $a0,blankSpace       #allocating text
113             syscall             #printing to the screen
114
115             li $v0, 1           #setting service
116             move $a0, $t6           #allocating value of counter ($t6)
117             syscall             #printing to the screen
118
119             li $v0,4            #setting service
120             la $a0,tabulator        #allocating text
121             syscall             #printing to the screen
122
123             li $v0, 1           #setting service
124             move $a0, $t4           #allocating value of array ($t4) at given index ($t0)
125             syscall             #printing to the screen
126
127             li $v0,4            #setting service
128             la $a0,tabulator        #allocating text
129             syscall             #printing to the screen
130
131             j displayingIndicator
132
133         ################ Displaying '#' represenattion through iteration #################
134         displayingIndicator:
135             blez $t4, newLine       #if value of counter ($t4) <= 0 go to 'loop'
136             li $v0,4            #setting service
137             la $a0,sign         #allocating text
138             syscall             #printing to the screen '#' sign
139             subu $t4,$t4, 5         #counter decrase by 5 (scale down by factor x5)
140
141             j displayingIndicator
```

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.).

## 7. Calculating average

```
143         ################ Calculating Average #################
144         calcAvg:
145             div $t9, $t6            #calculating average
146             mflo $t0            #moving result of average from $LO into $t0
147             j displayingStats
```

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.

## 8. Displaying data: analysis

```
155
156         ################ Displaying Max, Min and AVG #################
157         displayingStats:
158             li $v0,4            #setting service
159             la $a0,breakline        #allocating text
160             syscall             #printing to the screen
161
162             li $v0,4            #setting service
163             la $a0,max          #allocating text
164             syscall             #printing to the screen
165
166             li $v0, 1           #setting service
167             move $a0, $t7           #allocating value of max ($t7)
168             syscall             #printing to the screen
169
170             li $v0,4            #setting service
171             la $a0,blankSpace       #allocating text
172             syscall             #printing to the screen
173
174             li $v0,4            #setting service
175             la $a0,min          #allocating text
176             syscall             #printing to the screen
177
178             li $v0, 1           #setting service
179             move $a0, $t8           #allocating value of min ($t8)
180             syscall             #printing to the screen
181
182             li $v0,4            #setting service
183             la $a0,blankSpace       #allocating text
184             syscall             #printing to the screen
185
186             li $v0,4            #setting service
187             la $a0,avg          #allocating text
188             syscall             #printing to the screen
189
190             li $v0, 1
191             move $a0, $t0           #allocating value of average ($t0)
192             syscall             #printing to the screen
193
194             j end
195
196     ################ Terminating Program #################
197     end:
198         li $v0,10
199         syscall
```

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.

## 9. Exercises

```
15  .data
16  array:
17      .word 0x6E, 0x76, 0x80, 0x90, 0x85, 0x96, 0x8C, 0x93, 0xAC, 0x9D
18      .word 0xB2, 0xB3, 0xB2, 0xB7, 0xBE, 0xB3, 0xBF, 0xC3, 0xD0, 0xD3
19      .word 0xCB, 0xC4, 0xCA, 0xDC, 0xD5, 0xCA, 0xD5, 0xE3, 0xD6, 0xCF
20      #.word 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01 #enters min value of 1
21      .word 0xDB, 0xD1, 0xDE, 0xD0, 0xD7, 0xD6, 0xE3, 0xE2, 0xCA, 0xD2
22      .word 0xD8, 0xCF, 0xD3, 0xD1, 0xC9, 0xCC, 0xC9, 0xB2, 0xB8, 0xAF
23      .word 0xB9, 0xA1, 0xA3, 0xB0, 0x94, 0x95, 0x8E, 0x97, 0x95, 0x8B
24      .word 0x80, 0x7C, 0x84, 0x6F, 0x65, 0x70, 0x68, 0x5F, 0x54, 0x4F
25      #.word 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF #enters max value of 255
26      .word 0x49, 0x53, 0x4A, 0x4B, 0x36, 0x32, 0x3B, 0x2D, 0x2A, 0x33
27      .word 0x1E, 0x2D, 0x2A, 0x2A, 0x1D, 0x10, 0x10, 0x0A, 0x13, 0x09
28      .word 0x14, 0x16, 0x11, 0x0C, 0x18, 0x15, 0x13, 0x0F, 0x1A, 0x05
29      .word 0x05, 0x19, 0x15, 0x0F, 0x1A, 0x1B, 0x24, 0x25, 0x29, 0x27
30      .word 0x2C, 0x35, 0x2A, 0x38, 0x2E, 0x39, 0x47, 0x40, 0x54, 0x55
31
```

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).

```
16  array:
17      .word 0x6E, 0x76, 0x80, 0x90, 0x85, 0x96, 0x8C, 0x93, 0xAC, 0x9D
18      .word 0xB2, 0xB3, 0xB2, 0xB7, 0xBE, 0xB3, 0xBF, 0xC3, 0xD0, 0xD3
19      .word 0xCB, 0xC4, 0xCA, 0xDC, 0xD5, 0xCA, 0xD5, 0xE3, 0xD6, 0xCF
20      .word 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01 #enters min value of 1
21      #.word 0xDB, 0xD1, 0xDE, 0xD0, 0xD7, 0xD6, 0xE3, 0xE2, 0xCA, 0xD2
22      .word 0xD8, 0xCF, 0xD3, 0xD1, 0xC9, 0xCC, 0xC9, 0xB2, 0xB8, 0xAF
23      .word 0xB9, 0xA1, 0xA3, 0xB0, 0x94, 0x95, 0x8E, 0x97, 0x95, 0x8B
24      .word 0x80, 0x7C, 0x84, 0x6F, 0x65, 0x70, 0x68, 0x5F, 0x54, 0x4F
25      #.word 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF #enters max value of 255
26      .word 0x49, 0x53, 0x4A, 0x4B, 0x36, 0x32, 0x3B, 0x2D, 0x2A, 0x33
27      .word 0x1E, 0x2D, 0x2A, 0x2A, 0x1D, 0x10, 0x10, 0x0A, 0x13, 0x09
28      .word 0x14, 0x16, 0x11, 0x0C, 0x18, 0x15, 0x13, 0x0F, 0x1A, 0x05
29      .word 0x05, 0x19, 0x15, 0x0F, 0x1A, 0x1B, 0x24, 0x25, 0x29, 0x27
30      .word 0x2C, 0x35, 0x2A, 0x38, 0x2E, 0x39, 0x47, 0x40, 0x54, 0x55
31
```

Figure9a:

```
15  .data
16  array:
17      .word 0x6E, 0x76, 0x80, 0x90, 0x85, 0x96, 0x8C, 0x93, 0xAC, 0x9D
18      .word 0xB2, 0xB3, 0xB2, 0xB7, 0xBE, 0xB3, 0xBF, 0xC3, 0xD0, 0xD3
19      .word 0xCB, 0xC4, 0xCA, 0xDC, 0xD5, 0xCA, 0xD5, 0xE3, 0xD6, 0xCF
20      #.word 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01, 0x01 #enters min value of 1
21      .word 0xDB, 0xD1, 0xDE, 0xD0, 0xD7, 0xD6, 0xE3, 0xE2, 0xCA, 0xD2
22      .word 0xD8, 0xCF, 0xD3, 0xD1, 0xC9, 0xCC, 0xC9, 0xB2, 0xB8, 0xAF
23      .word 0xB9, 0xA1, 0xA3, 0xB0, 0x94, 0x95, 0x8E, 0x97, 0x95, 0x8B
24      .word 0x80, 0x7C, 0x84, 0x6F, 0x65, 0x70, 0x68, 0x5F, 0x54, 0x4F
25      .word 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF, 0xFF #enters max value of 255
26      #.word 0x49, 0x53, 0x4A, 0x4B, 0x36, 0x32, 0x3B, 0x2D, 0x2A, 0x33
27      .word 0x1E, 0x2D, 0x2A, 0x2A, 0x1D, 0x10, 0x10, 0x0A, 0x13, 0x09
28      .word 0x14, 0x16, 0x11, 0x0C, 0x18, 0x15, 0x13, 0x0F, 0x1A, 0x05
29      .word 0x05, 0x19, 0x15, 0x0F, 0x1A, 0x1B, 0x24, 0x25, 0x29, 0x27
30      .word 0x2C, 0x35, 0x2A, 0x38, 0x2E, 0x39, 0x47, 0x40, 0x54, 0x55
31
```

Figure9b:

```
                    Data Graph
-------------------------------------------------------------
| Item |  Value  |              Indicator                  |
-------------------------------------------------------------
    1   110     ######################
    2   118     ########################
    3   128     ##########################
    .   .       .
    29  214     ##########################################
    30  207     ########################################
    31  1       #
    32  1       #
    33  1       #
    34  1       #
    35  1       #
    36  1       #
    37  1       #
    38  1       #
    39  1       #
    40  1       #
    41  216     ##########################################
    42  207     ########################################
    .   .       .
    119 84      ################
    120 85      ################
-------------------------------------------------------------
MAX = 227    MIN = 1    AVG = 99
```

Figure10a:

```
                    Data Graph
-------------------------------------------------------------
| Item |  Value  |              Indicator                  |
-------------------------------------------------------------
    1   110     ######################
    2   118     ########################
    3   128     ##########################
    .   .       .
    69  84      ################
    70  79      ###############
    71  255     ##################################################
    72  255     ##################################################
    73  255     ##################################################
    74  255     ##################################################
    75  255     ##################################################
    76  255     ##################################################
    77  255     ##################################################
    78  255     ##################################################
    79  255     ##################################################
    80  255     ##################################################
    81  30      ######
    82  45      #########
    .   .       .
    119 84      ################
    120 85      ################
-------------------------------------------------------------
MAX = 255    MIN = 5    AVG = 133
```

Figure10b: