02/28/14 09:08:27 /Users/Clayton/Desktop/Repos/Comp-Org/Assignment 1/RieckSPIM1.a

```
#-------------------------------------------------------------+
# Author: Clayton Rieck                                       |
#-------------------------------------------------------------+
# DESCRIPTION:                                                |
# This program finds the MIN, MAX, MEAN and VARIANCE          |
# of given an array of up to 20 integers. It also displays    |
# a bar graph of frequencies of numbers in the array.         |
#-------------------------------------------------------------+
# INPUT:                                                      |
# The user will input up to 20 integers and will end          |
# input once a negative value is entered. The integers        |
# may not be 0 and greater than 15 (Range = [1,15])           |
#-------------------------------------------------------------+
# OUTPUT:                                                     |
# The program will output the MIN, MAX, MEAN, and VARIANCE    |
# of the array of integers and will also output a bar graph   |
# of the frequencies of numbers in the array                  |
#-------------------------------------------------------------+
    .data
myArray:    .word       0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
prompt:     .asciiz     "Please enter up to 20 positive integers and
a negative to end input"
var_div:    .asciiz     "VARIANCE: "
graph_div:  .asciiz     "BAR GRAPH"
endl:       .asciiz     "\n"
endl2:      .asciiz     "\n\n"
space:      .asciiz     ": "
space2:     .asciiz     ":  "
complete:   .asciiz     "Program complete"
array_err:  .asciiz     "Out of 1-15 range. Terminating program"
min:        .asciiz     "MIN: "
max:        .asciiz     "MAX: "
mean:       .asciiz     "MEAN: "
    .text
    .globl main

main:

    # ask user to input numbers
    la      $a1,    myArray             # load pointer into a1
    la      $a0,    prompt              # load prompt into a0

    li      $v0,    4                   # print out prompt
    syscall

    la      $a0,    endl
    li      $v0,    4
    syscall

    li      $t0,    0                   # MIN
    li      $t1,    0                   # MAX
    li      $t2,    0                   # ARRAY OFFSET
    li      $t3,    0                   # ARRAY LENGTH
    li      $t5,    0                   # SUM
```

```
 53
 54  #------------------------------------------------------
 55  # Starts the input prompt and stores numbers in array \
 56  # while also computing/finding the MIN, MAX, SUM and   \
 57  # ARRAY LENGTH                                         \
 58  #------------------------------------------------------
 59  gather_numbers:
 60
 61      bgt     $t3,    20,     three_ms       # if we've filled our list
 62
 63      li      $v0,    5                      # read integer into $v0
 64      syscall
 65
 66      move    $t4,    $v0                    # $t4 holds the value that
     the user inputted
 67
 68      beq     $t4,    0,      array_error # if enter a number less than
     1
 69      bgt     $t4,    15,     array_error # if enter a number above 15
 70
 71      bltz    $t4,    three_ms               # if enter negative then stop
     populating list
 72
 73      sw      $t4,    myArray($t2)           # store current element in
     $t4
 74      add     $t3,    $t3,    1              # add 1 to array length
 75
 76      add     $t5,    $t5,    $t4            # add input number to sum
 77
 78      add     $t2,    $t2,    4              # go to next index in array
 79
 80      beqz    $t0,    set_min                # initialize min with first
     number entered
 81
 82      blt     $t4,    $t0,    set_min     # if input less than MIN, set
     new min
 83
 84  #------------------------------------------------------
 85  # Checks to see if input is bigger than current MAX   \
 86  #------------------------------------------------------
 87  check_max:
 88      bgt     $t4,    $t1,     set_max
 89      j       gather_numbers
 90
 91  #------------------------------------------------------
 92  # Sets current MAX to inputted number                 \
 93  #------------------------------------------------------
 94  set_max:
 95      move     $t1,    $t4
 96      j        gather_numbers
 97
 98  #------------------------------------------------------
 99  # Checks to see if input is smaller than current MIN  \
100  #------------------------------------------------------
101  set_min:
102      move     $t0,    $t4
```

```
103     j        check_max
104
105 #-----------------------------------------------------
106 # Prints out the MAX, MIN, and MEAN of the inputted   \
107 # integers in the array                                \
108 #-----------------------------------------------------
109 three_ms:
110
111     la      $a0,    endl
112     li      $v0,     4
113     syscall
114
115     # print out MIN
116     la      $a0,    min
117     li      $v0,     4
118     syscall
119
120     li      $v0,     1
121     addi    $a0,    $t0,    0
122     syscall
123
124     la      $a0,    endl
125     li      $v0,    4
126     syscall
127
128     # print out MAX
129     la      $a0,    max
130     li      $v0,    4
131     syscall
132
133     li      $v0,    1
134     addi    $a0,    $t1,     0
135     syscall
136
137     # print out MEAN
138     la      $a0,    endl
139     li      $v0,     4
140     syscall
141
142     la      $a0,    mean
143     li      $v0,     4
144     syscall
145
146     # calculates and then prints mean
147     li      $v0,    1
148     div     $t5,    $t3             # divide sum of numbers by length
    of array
149     mflo    $a0
150     syscall
151
152     la      $a0,    endl2
153     li      $v0,    4
154     syscall
155
```

```
156 # ---------------------------------------------------------
157 # Start Standard Deviation
158
159     li      $t2,    0               # array offset
160     li      $t6,    1               # current index in the array
161     li      $s0,    0               # result of summation
162     li      $s3,    1               # CONSTANT 1 (used for 1/(n-1))
163
164     div     $t5,    $t3             # calculate x bar (mean)
165     mflo    $t0                     # store mean in $t0
166
167 #---------------------------------------------------------
168 # Loops through array and subtracts each element by   \
169 # the MEAN and then sqaures the result                 \
170 #---------------------------------------------------------
171 sum:
172
173     bgt     $t6,    $t3, variance   # if current index is greater
    than length of list
174
175     lw      $t4,    myArray($t2)    # load current element
176
177     sub     $t1,    $t4,    $t0     # subtract list element by mean
178     mul     $t1,    $t1,    $t1     # square difference
179     add     $s0,    $s0,    $t1     # add terms of summation
180
181     add     $t6,    $t6,    1       # increment index by 1
182     add     $t2,    $t2,    4       # increment offset by 4
183     j       sum
184
185 #---------------------------------------------------------
186 # Takes the result of the summation (from above) and  \
187 # multiplies that by 1/(n-1) giving us s^2 (VARIANCE)  \
188 #---------------------------------------------------------
189 variance:
190     sub     $s1,    $t3,    1       # get n - 1
191     div     $s0,    $s1             # do sum/(n-1)
192     mflo    $s1                     # store answer in $s1
193
194     # print out variance
195     la      $a0,    var_div
196     li      $v0,    4
197     syscall
198
199     move    $a0,    $s1             # move vairance to $a0 for
    printing
200     li      $v0,    1
201     syscall
202
203     la      $a0,    endl2
204     li      $v0,    4
205     syscall
206
207 # ---------------------------------------------------------
208 # Start Bar Graph
209
```

```
210 #-------------------------------------------------------
211 # Initializes counters for the bar graph (range number \
212 # and offset constant for finding end index of the array\
213 #-------------------------------------------------------
214 set_graph_counter:
215     la      $a0,    graph_div
216     li      $v0,    4
217     syscall
218 
219     la      $a0,    endl
220     li      $v0,    4
221     syscall
222 
223     li      $t6,    1               # keep counter for bar graph
    range
224     li      $s0,    4               # used as constant for finding
    end index of array
225 
226 #-------------------------------------------------------
227 # Does 1 iteration through the array for each number  \
228 # the 1-15 range our integers are within               \
229 #-------------------------------------------------------
230 graph_loop:
231 
232     bgt     $t6,    15,     finish  # if at end of our range
233 
234     li      $t2,    0               # reset array index counter to 0
235     li      $t7,    0               # keep track of number of
    ocurrences
236 
237     # print out number in range
238     li      $v0,    1
239     add     $a0,    $t6,    0
240     syscall
241 
242     blt     $t6,    10,      double_space # if number in range > 9
243     # print a ': ' after number in range
244     la      $a0,    space
245     li      $v0,    4
246     syscall
247     j       set_graph
248 
249 #-------------------------------------------------------
250 # Formats bar graph output to do 2 spaces so that     \
251 # the bars line up for the single digits and double    \
252 # digits                                               \
253 #-------------------------------------------------------
254 double_space:
255     la      $a0,    space2
256     li      $v0,    4
257     syscall
258 
259 #-------------------------------------------------------
260 # Conducts checks on if a number was encountered in   \
261 # the array and if on the last index of the array.     \
```

```
262 # Also jumps to beginning of loop                          \
263 #-------------------------------------------------------
264 set_graph:
265
266     lw      $t4,    myArray($t2)    # load current element
267
268     beq     $t4,    $t6,    add_bar # if current item in array equals
    number
269                                     # we're checking go to add_bar
    label
270     add     $t2,    $t2,    4       # loop through list
271
272     div     $t2,    $s0             # divide the array offset by 4 to
    get current index number
273     mflo    $s1                        # move quotient to $s1
274     beq     $s1,    $t3,    increment_number # if the array length
    and quotient are
275                                             # equal then we are at
    the end of the list and
276                                             # need to start again
277     j       set_graph
278
279 #-------------------------------------------------------
280 # Outputs a 0 next to a number denoting that it was   \
281 # found                                                \
282 #-------------------------------------------------------
283 add_bar:
284
285     # print out 0 for bar
286     li      $v0,    1
287     li      $a0,    0
288     syscall
289
290
291     add     $t2,    $t2,    4       # loop through list
292
293     # check to see if at end of the list (same as above)
294     div     $t2,    $s0
295     mflo    $s1
296     beq     $s1,    $t3,    increment_number
297
298     j       set_graph
299
300 #-------------------------------------------------------
301 # Increments the range number by 1                     \
302 #-------------------------------------------------------
303 increment_number:
304
305     addi    $t6,    $t6,    1        # increment range number by 1
306
307     # new line for new bar
308     la      $a0,    endl
309     li      $v0,    4
310     syscall
```

```
312     j       graph_loop
313
314 #-----------------------------------------------------
315 # Outputs a program completed message                 \
316 #-----------------------------------------------------
317 finish:
318
319     la      $a0,    endl2           # 2 new line characters
320     li      $v0,    4
321     syscall
322
323     la      $a0,    complete        # ending message
324     li      $v0,    4
325     syscall
326
327     li      $v0,    10
328     syscall
329
330 #-----------------------------------------------------
331 # Displays an error when user enters number outside   \
332 # of 1-15 range                                        \
333 #-----------------------------------------------------
334 array_error:
335     la      $a0,    endl
336     li      $v0,    4
337     syscall
338
339     la      $a0,    array_err
340     li      $v0,    4
341     syscall
342
343     li      $v0,    10
344     syscall
```