Summer 2021

California State University, Northridge

Department of Electrical & Computer Engineering

MINI project3

July 12, 2021

Ryan Mark

```
1   ; Ryan mark
2   ; mini project 3
3                   AREA mini3, CODE, READONLY
4   rows            EQU   16 ; amount of rows in image
5   collum          EQU   16    ; amount of collums in  image
6   storage         EQU   0x40000000 ;adress of the unfiltered image
7   filter_image    EQU   0x40000100 ;the adres of the image that is going to be filtered
8   Histo1          EQU   0x40000200 ;stores data for the histogram of unfiltered image
9   Histo2          EQU   0x40000300 ; stores data for histogram of filtered image
10  intensity       EQU   0x40000400 ; store the the list inntenitys found in unfiltered image
11  filt_intensity  EQU   0x40000500 ; store the list of intensity found in filtered immage
12
13                  ENTRY
14                  MOV r2 ,#collum ;counter for columns
15                  MOV r3,#storage
16  Image           DCD row1,row2,row3,row4,row5,row6,row7,row8,row9,row10,row11,row12,row13,row14,row15,
    row16; array of all the rows
17                  ALIGN
18
19                  LDR r0,= Image  ;loads adress of the 2d image
20
21  ;----------------------2d map to
    1d-----------------------------------------------------------------
22  ONE_D           LDR r4,[r0],#4
23                  MOV r1,#rows ; counter for elements with thin a row
24  ELEMENT         LDRB r5,[r4],#1
25                  STRB r5,[r3],#1
26                  SUBS r1,r1,#1; counter goes down
27                  BNE ELEMENT
28                  SUBS r2,r2,#1
29                  BNE ONE_D
30  ;----------------------1d to 2d
    map-----------------------------------------------------------------------
31                  LDR r0,= filter_image;image that will be filtered later on
32                  MOV r2, #collum;total number of number in the 16x16 matrix
33                  MOV r3,#storage
34
35                  MOV r1,#256 ; counter for elements
36  COPY            LDRB r4,[r3],#1;
37                  STRB r4,[r0],#1; create duplitcae of input that will be later fileter as the output
38                  SUBS r1,r1,#1
39                  BNE COPY
40  ;-----------------------------------------------------histogram for unfiltered
    image-------------------------------------------------------
41
42                  LDR r0,=Histo1; will contain data for 1st histogram
43                  MOV r1,#0 ; counter used to co mpare vaules in the image
44                  MOV r4,#256 ;hiogram loop counter
45                  LDR r7, =intensity
46  HISTO           MOV r2, #0; counter for how many of what pixel intesity is in the  image in
47                  MOV r3, #256 ;total number of element in the 16x16 matrix
48                  LDR r5, =storage; get adress of the one dimensional array used to compare and also
    reset r5 so it can be used again
49  ;-------------------------------------array number search----------------------
50  search          LDRB r6,[r5],#1
51                  CMP  r6,r1 ; compare the value of the element to the number that is being searched for
52                  ADDEQ r2,r2,#1; add to counter if one of the element is equal to the number being
    compared too
53                  SUBS r3,r3,#1
54                  BNE search
55  ;-----------------------------end of array number search loop----------------
56                  CMP  r2,#0 ;check if r2  has any number of a specific pixel intensity
57                  STRBNE r2 ,[r0],#1; stores counter of the number of a sepcific pixel intensity if
    couter is not zero
58                  CMP r2,#0
59                  STRBNE r1,[r7],#1 ; store the intensity that exists in the image
60                  ADD r1,r1,#1; set the next number that going to be searched and comapered with
61                  SUBS r4,r4,#1 ; histogram counter goes down
62                  BNE HISTO
63  ;-------------------------------------------------------------image
    filter--------------------------------------
64                  LDR r0, =Image ; the original image
```

```
65                  LDR r1, =filter_image ;  the image that will be filtered
66                  ADD r1,r1,#17 ; goes to the adress row 2 colum 2 whul will be the first pixel filtered
67                  MOV r12,#14; counter for rows  when excluding row 1 and 16
68   ;-------------------------------------------------------  image filter
     loop---------------------------------------
69   image_filter
70                  LDR r3,[r0,#4];goes to row 2 by getttin the adress of row2 pre indexing used to not
     change r0
71                  LDR r4,[r0,#8];goes to row 3 by getttin the adress of row3 pre indexing used to not
     change r0
72                  LDR r2,[r0],#4 ; goes to row 1 by getttin the adress of row post indexing used to
     change r0 by adding 4 to adress to shift the starting poiint down by 1 row for the next cycle
73
74                  MOV r13,#14 ; this will serve as the collum counter since we do not filter colum 1 or 16
75   ; ----------------pixel calculation assuming 3x3 matrix use for
     caluculation-------------------------------
76   pixel_filter    MOV r9,#0; value for filter of a pixel
77                  LDRB r7,[r2,#1]; row 1 collumn 2 value
78                  LDRB r8,[r2,#2]; row 1 collum 3 value a
79                  LDRB r6,[r2],#1; row 1 collum 1 value  also adds one to the adress so we go through all
     collums
80
81                  ; we do not multiply (row1,collum1) since it being multiplied by one
82                  MOV r7,r7,LSL #1; multiply (row1,collum 2) by 2
83                  ; we do not multiply (row1,collum3) since it being multiplied by one
84
85                  ADD r7 ,r6,r7;  (row1,collum1) + (row1,collum2)
86                  ADD r8,r7,r8   ; (row1,collum1) + (row1,collum2)+ (row1, collum3)
87                  ADD r9,r9,r8     ;0 +(row1,collum1) + (row1,collum2)+ (row1, collum3)
88
89                  LDRB r7,[r3,#1]; row 2 collumn 2 value
90                  LDRB r8,[r3,#2]; row 2 collum 3 value
91                  LDRB r6,[r3],#1; row 2 collum 1 value  also adds one to the adress so we go through all
     collums
92
93                  MOV r6,r6,LSL #1 ;multiply (row2,colum1) by 2
94                  MOV r7,r7,LSL #2; multiply(row2,collum2) by 4
95                  MOV r8,r8,LSL #1 ; multiply ((row2,collum2) by 2
96
97                  ADD r7 ,r6,r7;  (row2,collum1) + (row2,collum2)
98                  ADD r8,r7,r8  ; (row2,collum1) + (row2,collum2)+ (row2,collum3)
99                  ADD r9,r9,r8  ; (row1,collum1) + (row1,collum2)+ (row1, collum3)+(row2,collum1) +
     (row2,collum2)+ (row2,collum3)
100
101                 LDRB r7,[r4,#1]; row 3 collumn 2 value
102                 LDRB r8,[r4,#2]; row 3 collum 3 value a
103                 LDRB r6,[r4],#1; row 3 collum1 value  also adds one to the adress so we go through all
     collums
104
105                 ; we do not multiply (row3,collum1) since it being multiplied by one
106                 MOV r7,r7,LSL #1; multiply by 2
107                 ; we do not multiply (row3,collum3) since it being multiplied by one
108
109                 ADD r7 ,r6,r7; (row3,collum1) + (row3,collum2)
110                 ADD r8,r7,r8 ;(row3,collum1) + (row3,collum2)+ (row3,collum3)
111                 ADD r9,r9,r8 ;(row1,collum1) + (row1,collum2)+ (row1, collum3)+(row2,collum1) +
     (row2,collum2)+ (row2,collum3) +(row3,collum1) + (row3,collum2)+ (row3,collum3)
112
113                 MOV r9,r9,LSR #4 ; divide by 16 by shifting binary form by 4 to the right
114                 STRB r9,[r1],#1; store new filer value in designated destination and shiftes the pixel
     filter by one
115                 SUBS r13,r13,#1
116                 BNE pixel_filter
117  ;-------------------------------------------------end of pixel fliter loop-----------------------
118                 ADD r1,r1,#2; makes it so r1 has the adress of the next row while also makeing sure it
     starts at collum 2 of the next row
119                 SUBS r12,r12,#1 ; row counter goes down by 1
120                 BNE image_filter
121  ;------------------------------------------------ end of image filter loop-------------------------
122
123
124  ;-----------------------------------------------------------histogram for filtered
```

```
      image----------------------
125
126                 LDR r0,=Histo2; will contain data for 1st histogram
127                 MOV r1,#0 ; counter used to co mpare vaules in the image
128                 MOV r4,#256 ;hiogram loop counter
129                 LDR r7, =filt_intensity
130   HISTO_FILTER  MOV r2, #0; counter for how many of what pixel intesity is in the  image in
131                 MOV r3, #256 ;total number of element in the 16x16 matrix
132                 LDR r5, =filter_image; get adress of the one dimensional array used to compare
133   ;-----------------------------------------array number search----------------------
134   search_filter LDRB r6,[r5],#1
135                 CMP  r6,r1
136                 ADDEQ r2,r2,#1; add to counter if one of the element is equal to the number being
      compared too
137                 SUBS r3,r3,#1
138                 BNE search_filter
139   ;-----------------------------end of array number search loop----------------
140                 CMP  r2,#0 ;check if r2  has any number of a specific pixel intensity
141                 STRBNE r2 ,[r0],#1; stores counter of the number of a sepcific pixel intensity if
      couter is not zero
142                 CMP r2,#0
143                 STRBNE r1,[r7],#1 ; store the intensity that exists in the image
144                 ADD r1,r1,#1; set the next number that going to be searched and comapered with
145                 SUBS r4,r4,#1 ; histogram counter goes down
146                 BNE HISTO_FILTER
147
148
149   stop          B stop
150   ;the  image
151   row1          DCB 0,0,0,0,255,255,255,255,0,0,0,0,255,255,255,255
152   row2          DCB 0,0,0,0,255,255,255,255,0,0,0,0,255,255,255,255
153   row3          DCB 0,0,0,0,255,255,255,255,0,0,0,0,255,255,255,255
154   row4          DCB 0,0,0,0,255,255,255,255,0,0,0,0,255,255,255,255
155   row5          DCB 255,255,255,255,0,0,0,0,255,255,255,255,0,0,0,0
156   row6          DCB 255,255,255,255,0,0,0,0,255,255,255,255,0,0,0,0
157   row7          DCB 255,255,255,255,0,0,0,0,255,255,255,255,0,0,0,0
158   row8          DCB 255,255,255,255,0,0,0,0,255,255,255,255,0,0,0,0
159   row9          DCB 0,0,0,0,255,255,255,255,0,0,0,0,255,255,255,255
160   row10         DCB 0,0,0,0,255,255,255,255,0,0,0,0,255,255,255,255
161   row11         DCB 0,0,0,0,255,255,255,255,0,0,0,0,255,255,255,255
162   row12         DCB 0,0,0,0,255,255,255,255,0,0,0,0,255,255,255,255
163   row13         DCB 255,255,255,255,0,0,0,0,255,255,255,255,0,0,0,0
164   row14         DCB 255,255,255,255,0,0,0,0,255,255,255,255,0,0,0,0
165   row15         DCB 255,255,255,255,0,0,0,0,255,255,255,255,0,0,0,0
166   row16         DCB 255,255,255,255,0,0,0,0,255,255,255,255,0,0,0,0
167
168
169                 END
```

START of the code

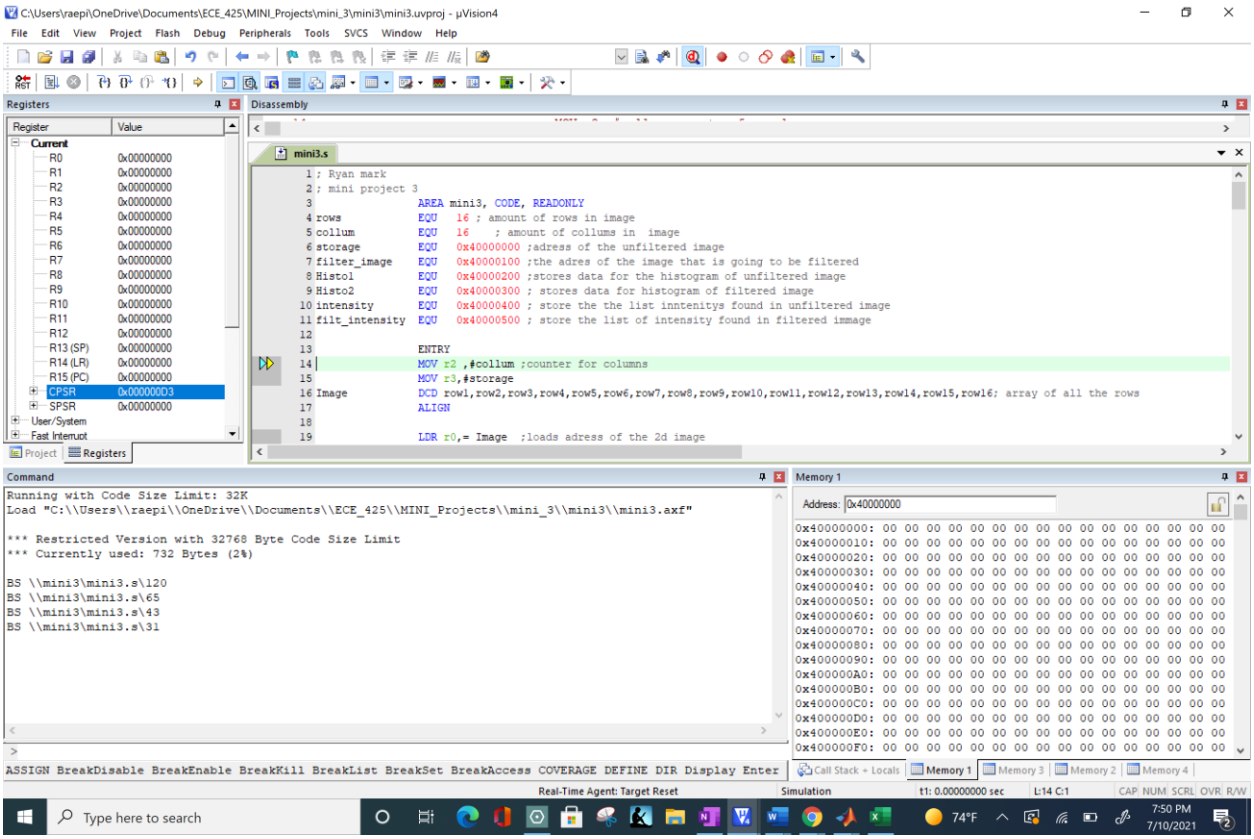
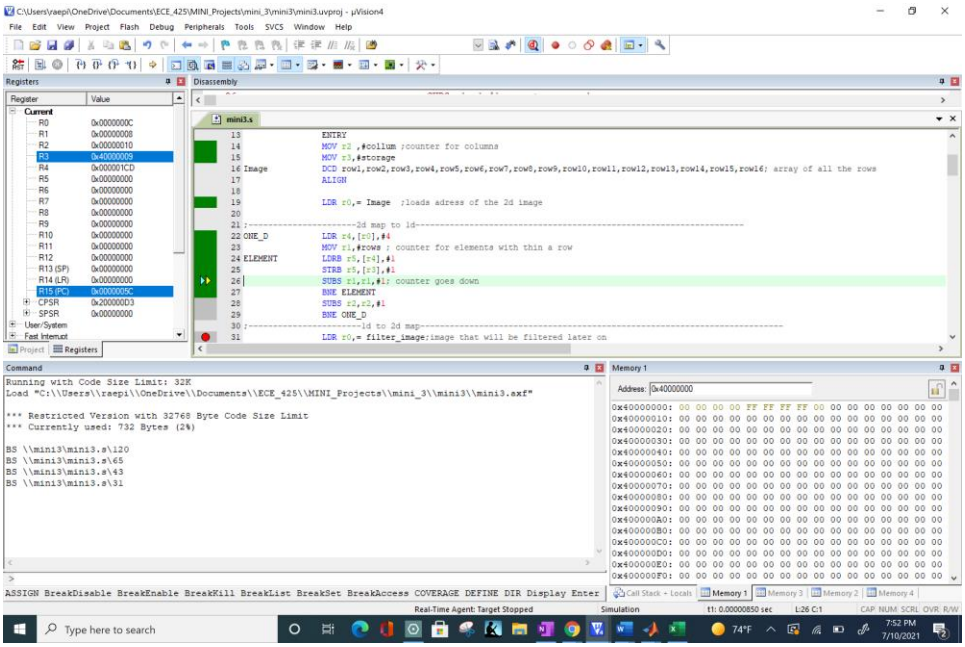


Figure 1 proof code is able to start

TASK 1

Store the image in the memory. You need to map the two dimensional image data to one dimensional array. You must be able to map the reverse process to find the two dimensional data from one dimensional array. What is the mapping formula for each case for this image clip? Can you present the mapping formula for a general case of MxN image?



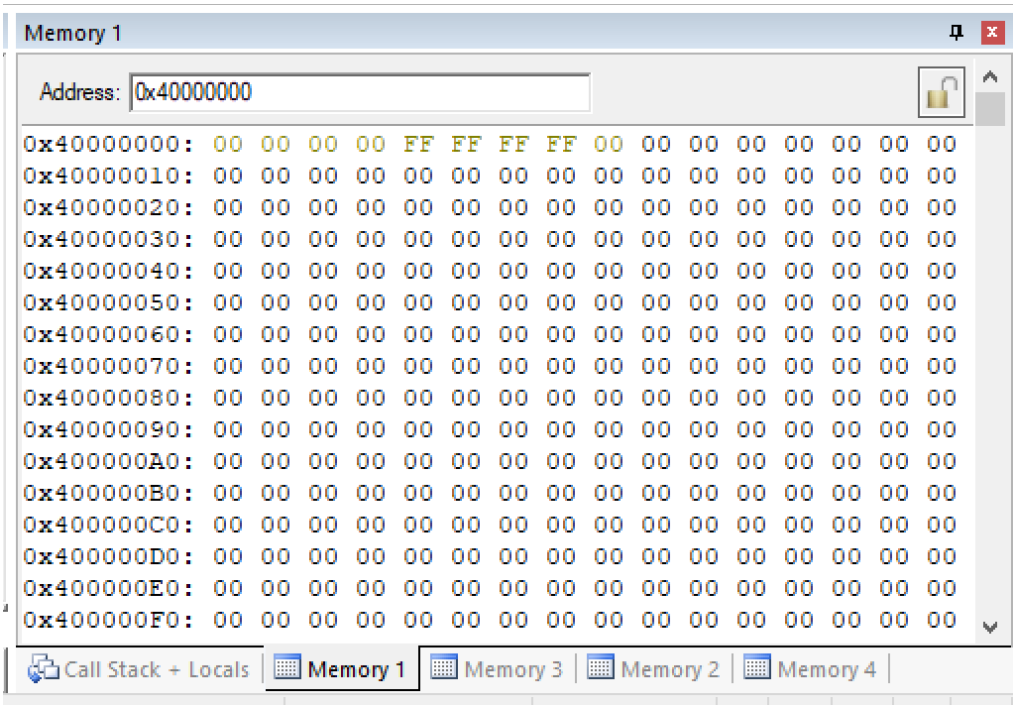
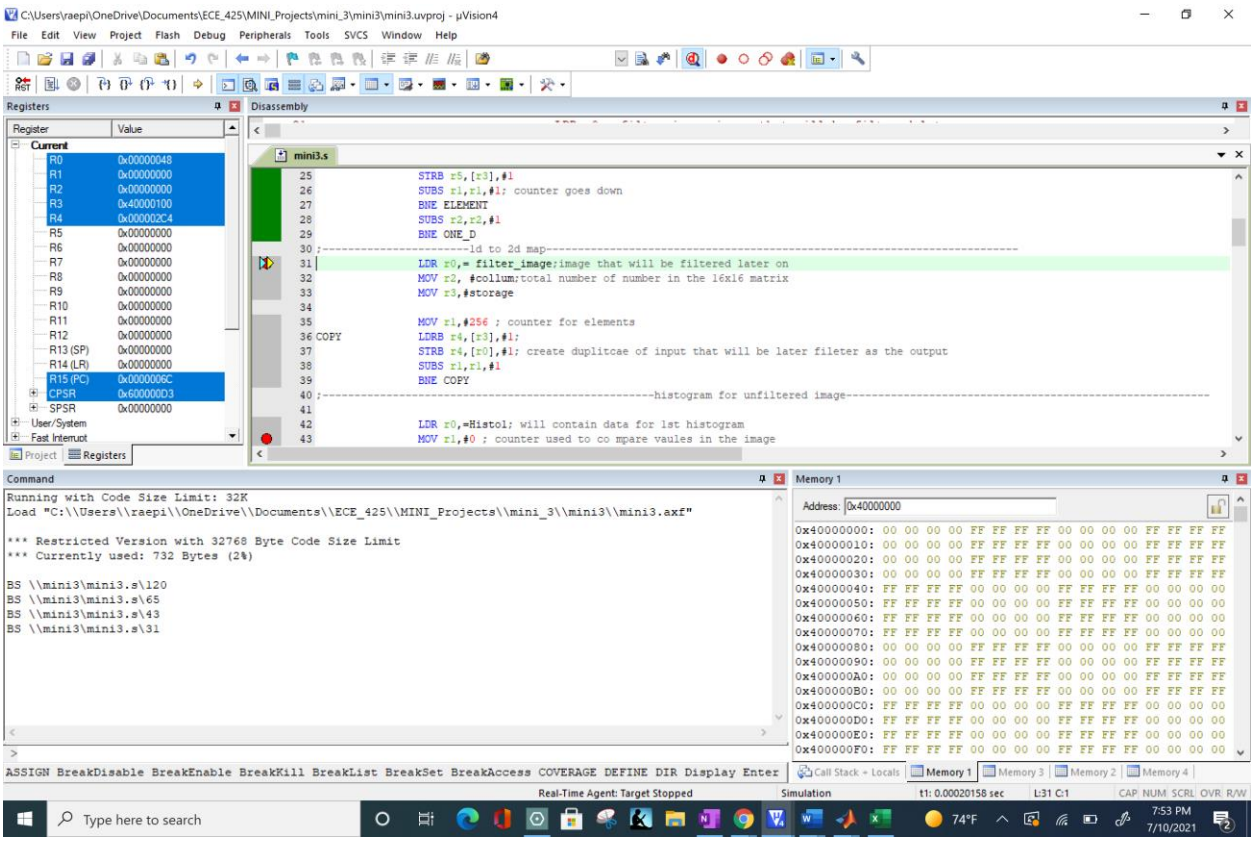


Figure 2: image in the process of being stored in adress 0x40000000



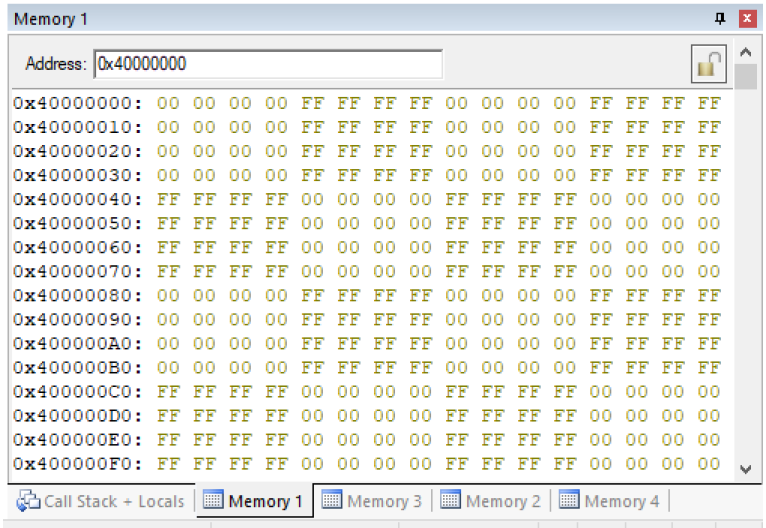
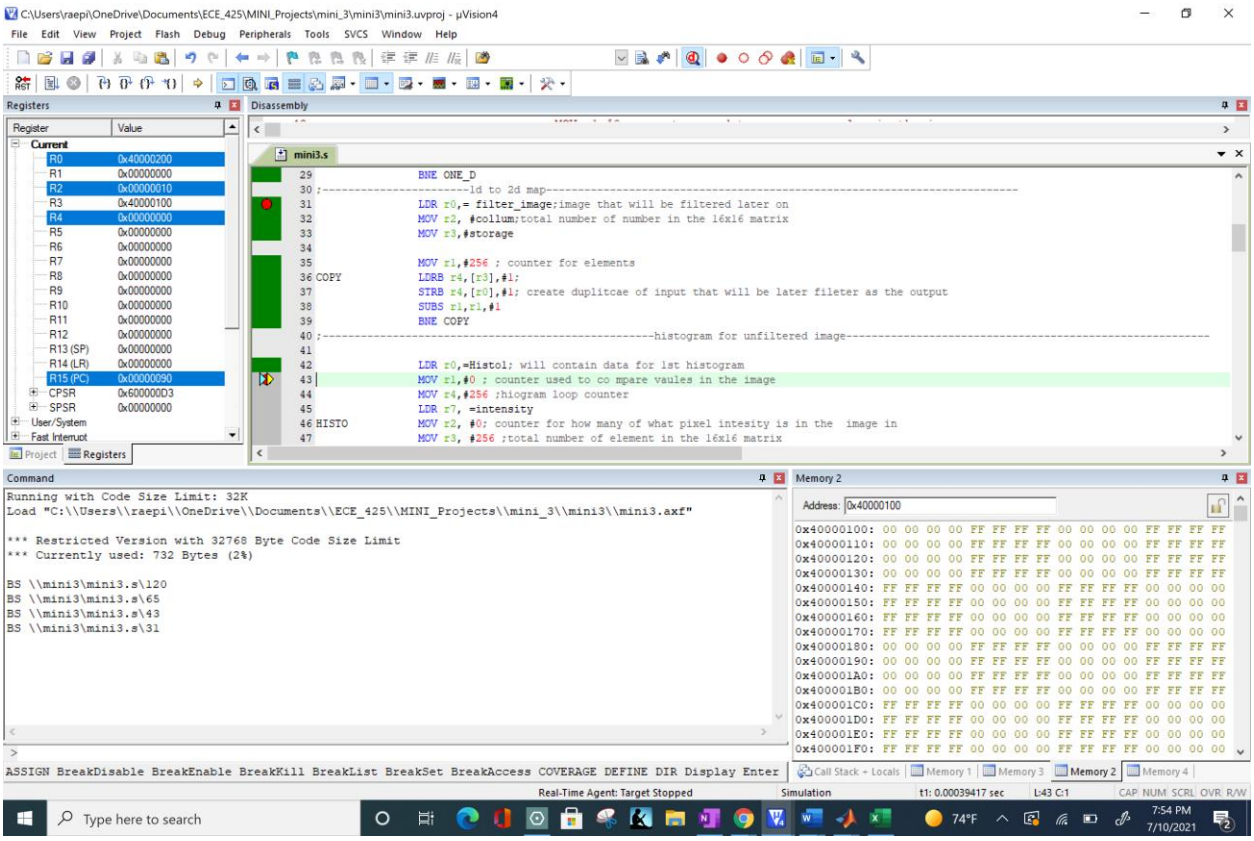
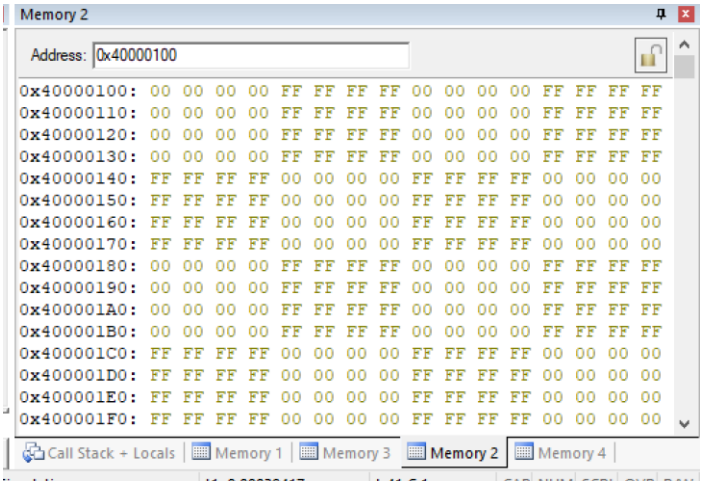


Figure 3 full image written into the address 0x40000000

For the mapping formula I take I make a nested loop to create the 1d array the main loop goes through the rows starting with row 1 and the sub loop goes through each element within the and the nested loop keeps cycling until it has gone through all 16 rows. For example on the first cycle we go through row one the sub loops cycles 16 times for each column to get all 16 element for a row.

Making a copy of the image into address 0x40000100 which will be the image I eventually filter





Task 2

Write an ARM assembly program to calculate the image histogram and use Excel to provide the histogram graph. Your histogram should display the pixel intensity values on x axis and the number of pixels with each intensity value on the y axis. For the given image, this is a trivial operation as there are only two intensities: 0 and 255. Your program should read one input array (image data) and output two arrays: pixel values (intensity values) and their corresponding number of pixels with that intensity value. For the input image, this would be 0 and 255 on the x axis and the number of pixels for each value (128 and 128) on the y axis

Basically, what I did to get the histogram data is I made a nested loop one to go through all 256 types of pixel intensity's and the sub loop goes through the image to see how many of that pixel intensity exists within the image. I f the intensity is found that intensity that exist is recorded into address 0x40000400. The amount of pixel that have the found intensity is recorded into address 0x40000200.Due to the nature of my method of getting the histogram it should be able to work with any image.

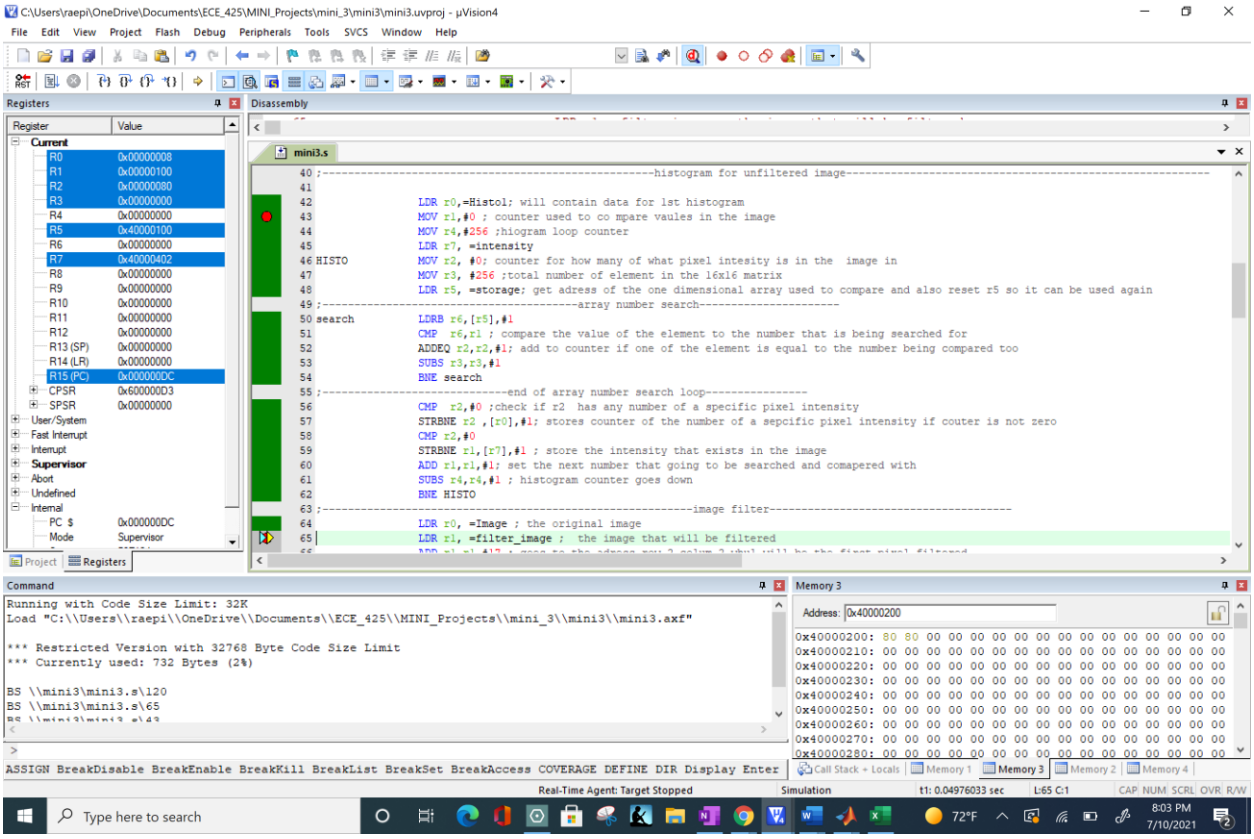


Figure 2.1 : Code for getting histogram data

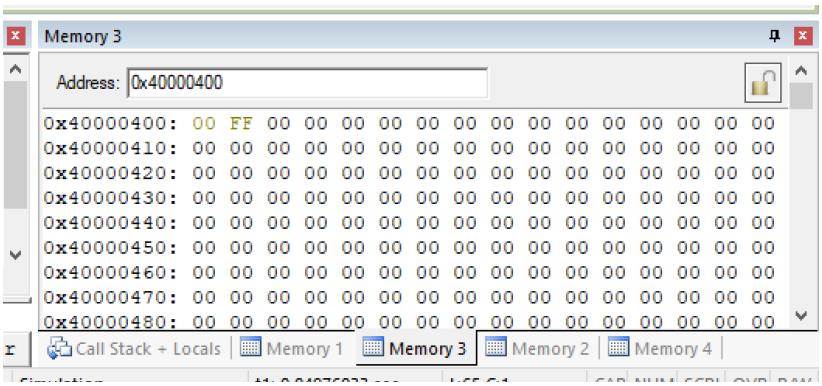


Figure 2.2 : intensity's found in unfiltered image stored in address 0x40000400. The intensity 0 and 255 exist for the u filtered image.

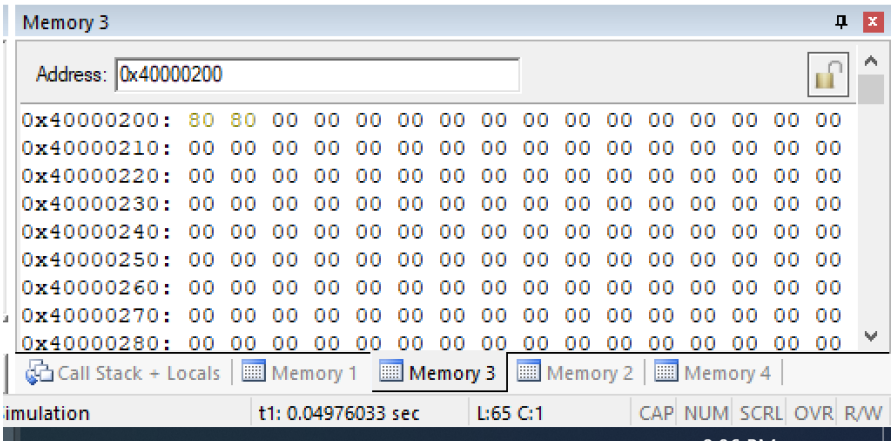


Figure 2.3 :Number of the found intensity for unfiltered image stored in address 0x40000200, Show 128 (80in hexadecimals) pixel had intensity 0 were found and 128 (80 in hexadecimals) pixel had intensity 255 were found

TASK 3

3. Write an ARM assembly program to apply image averaging filter to the original given image. Your program should read all image pixel values and apply the averaging filter to find the new values. Your program should process all image data in a loop to produce filtered image. Store the filtered image at different address

This part of the project was probly the most challenging part. I was able to complete this task by using the original image to help make calculation for every filtered pixel and store them into the duplicate image I made earlier. That way the filtered image is in a completely different address than the original. I crated the filter image using a nested loop I made use of pre and post indexing so I could access 3 rows at a time while also only shifting down 1 row as oppose to 3 rows that way everything get filtered in the end with the exception of row 1 and 16.For example the 1st cyle uses rows 1, 2,3 to filter row 2 and for the 2nd cycle it will use rows 2,3,4 to filter row 3. The pixel calculation loop goes through all column for

the row that being filter and like the main loop it makes use of pre and post indexing to get 3 element in a single row while only shifting by 1 after getting 3 elements.

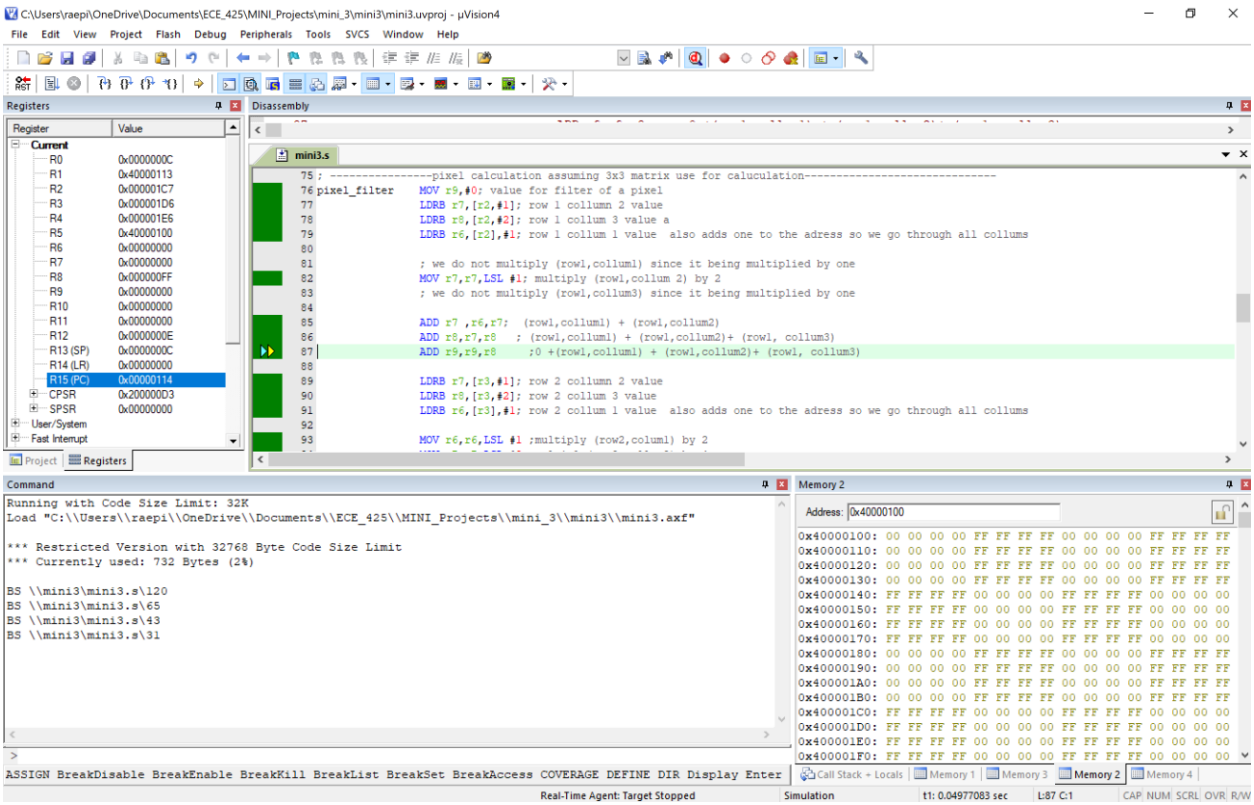
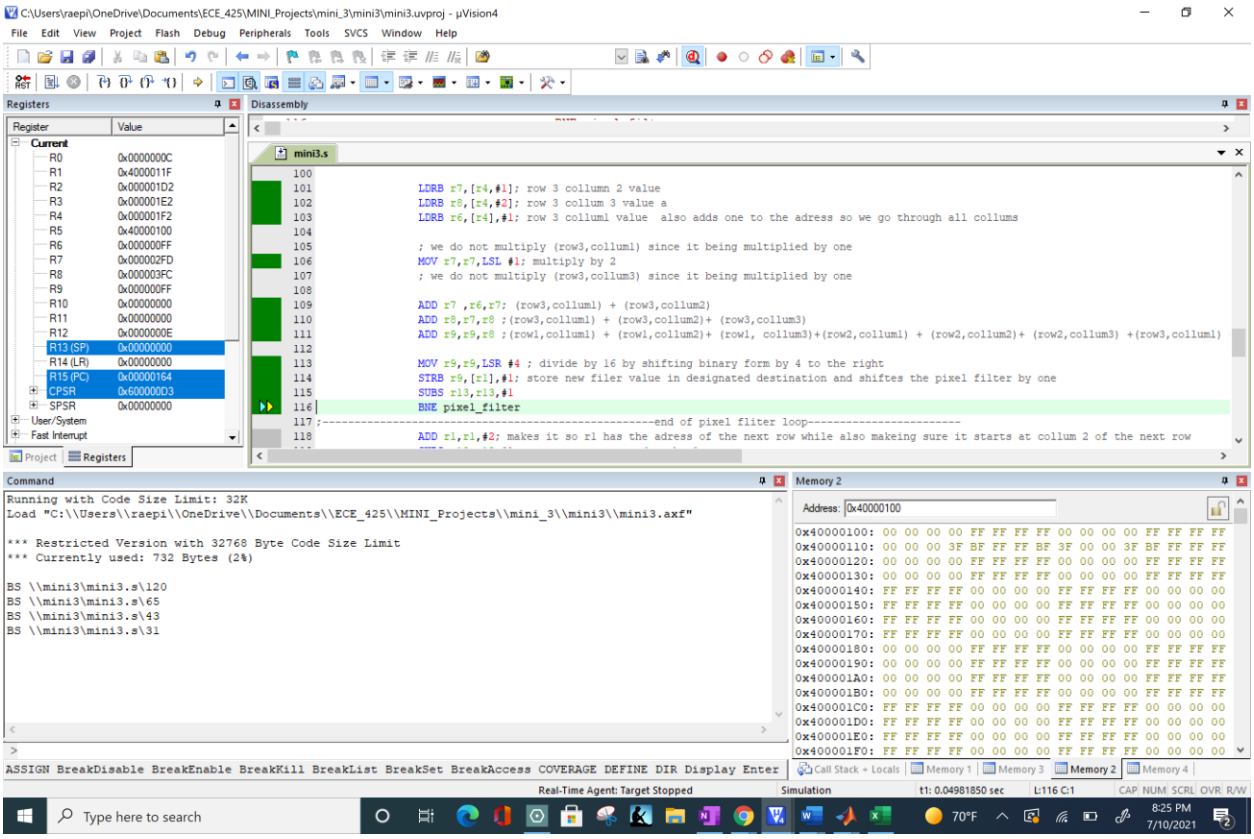


Figure 3.1 : calculating filtered pixels for the 2nd row



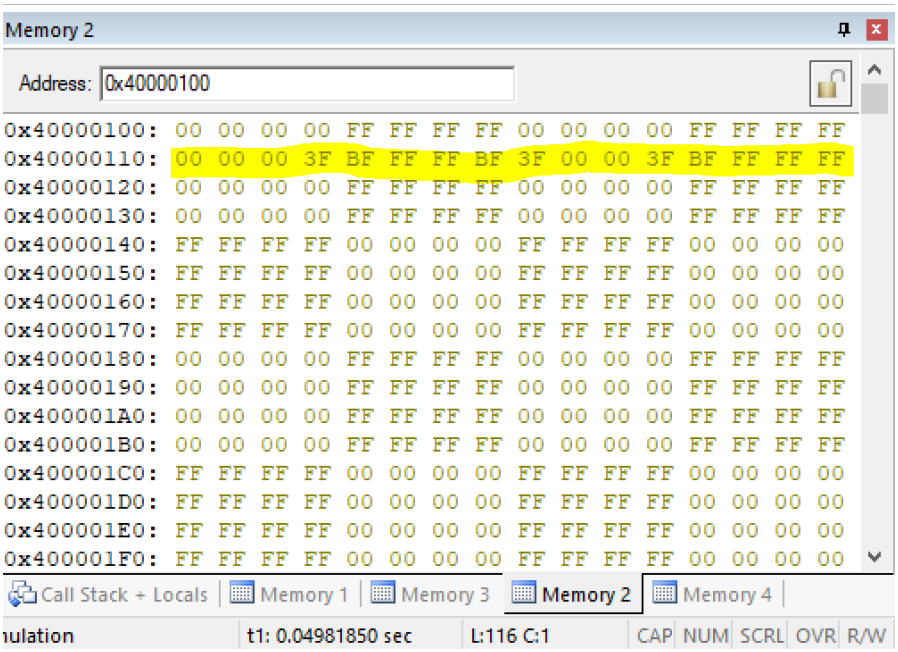
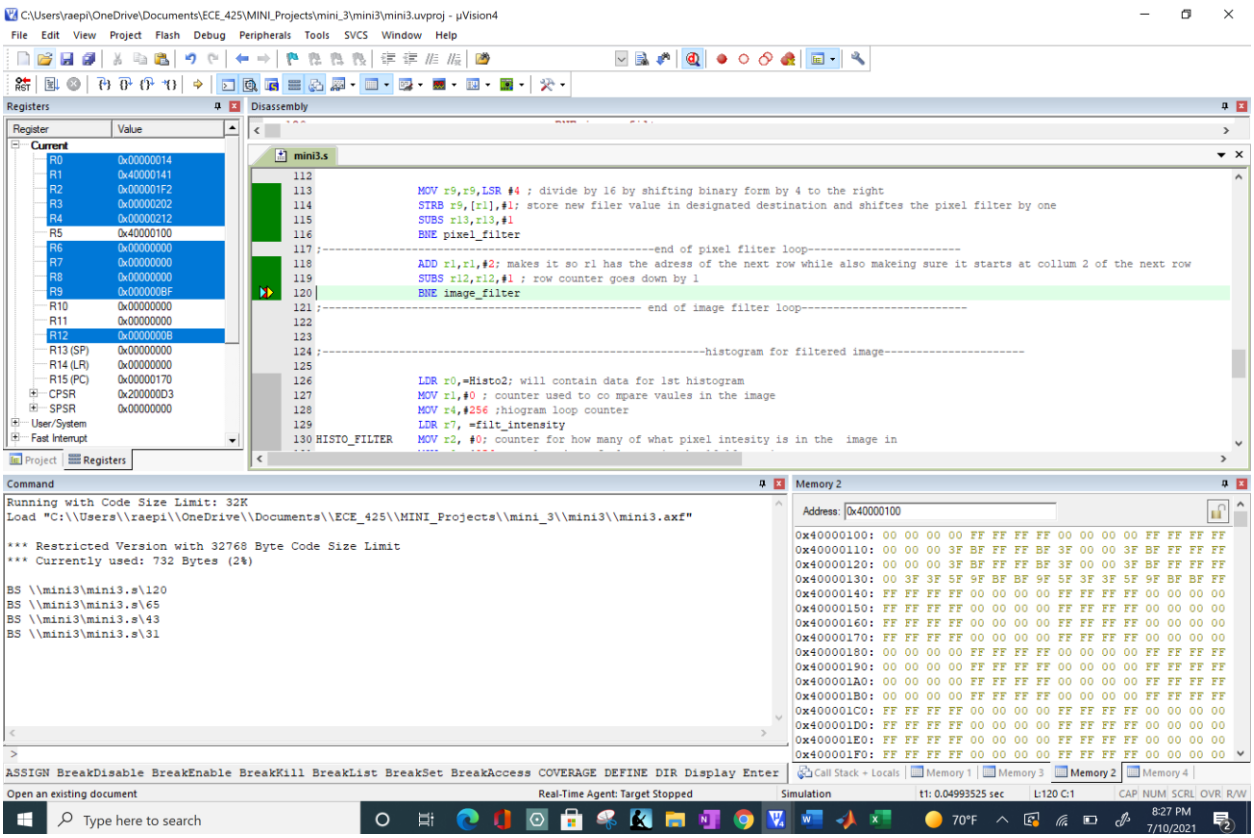


Figure 3.2 : row 2 has been filtered



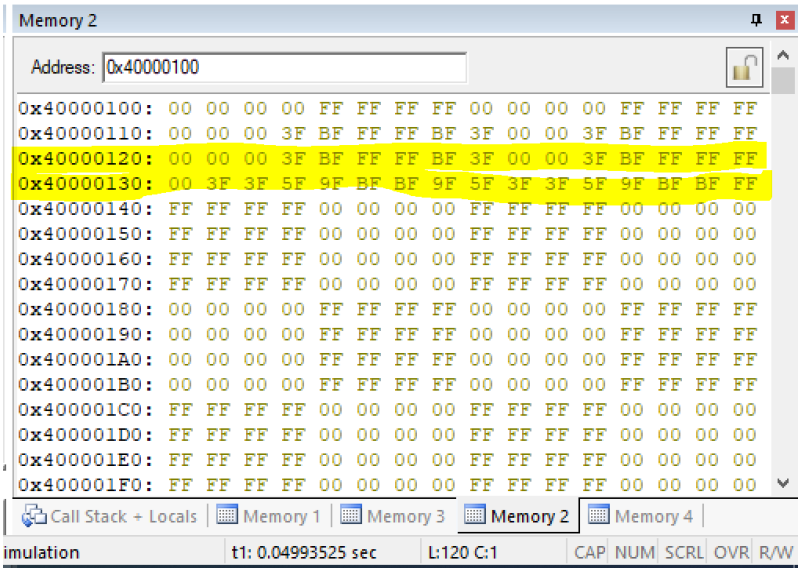
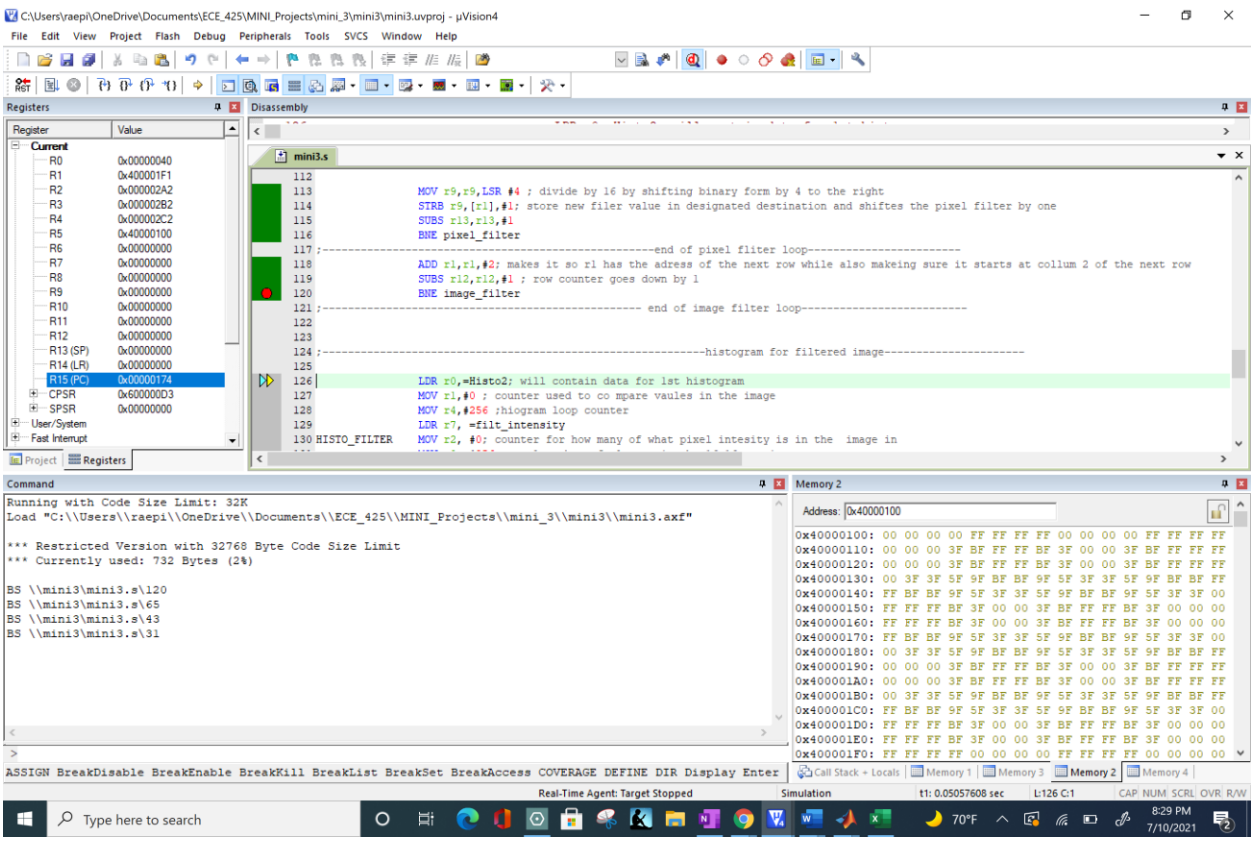


Figure 3.3 row 3 and 4 filtereed



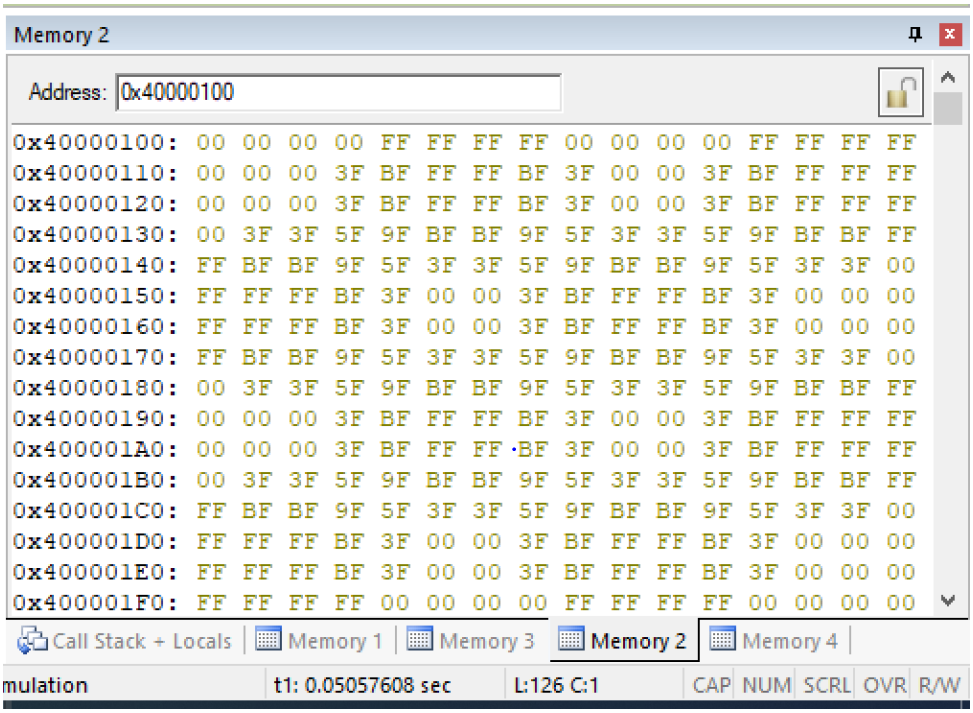
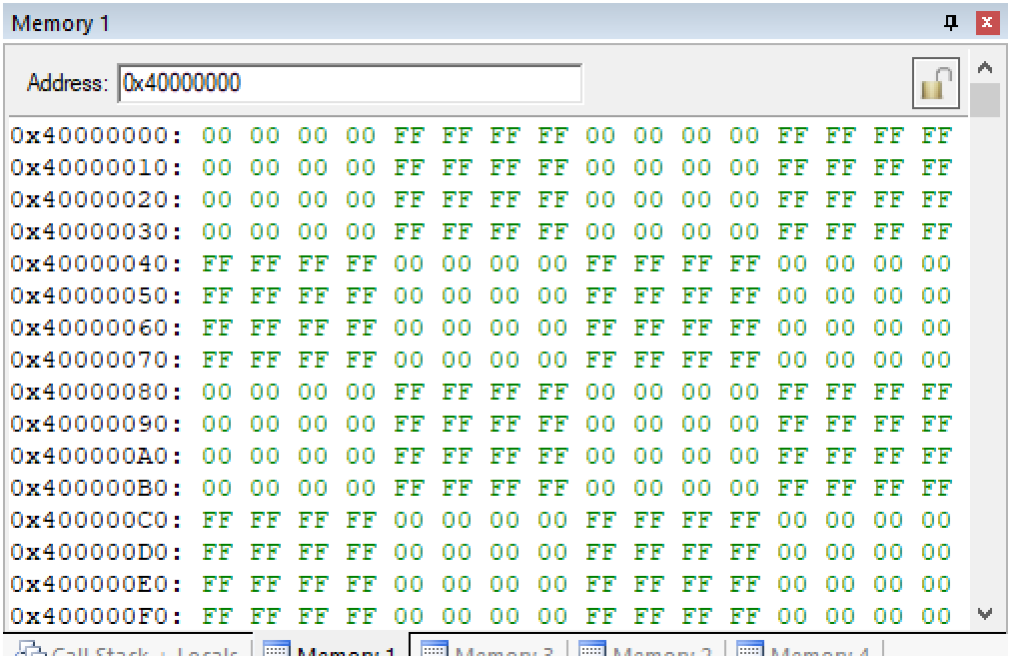
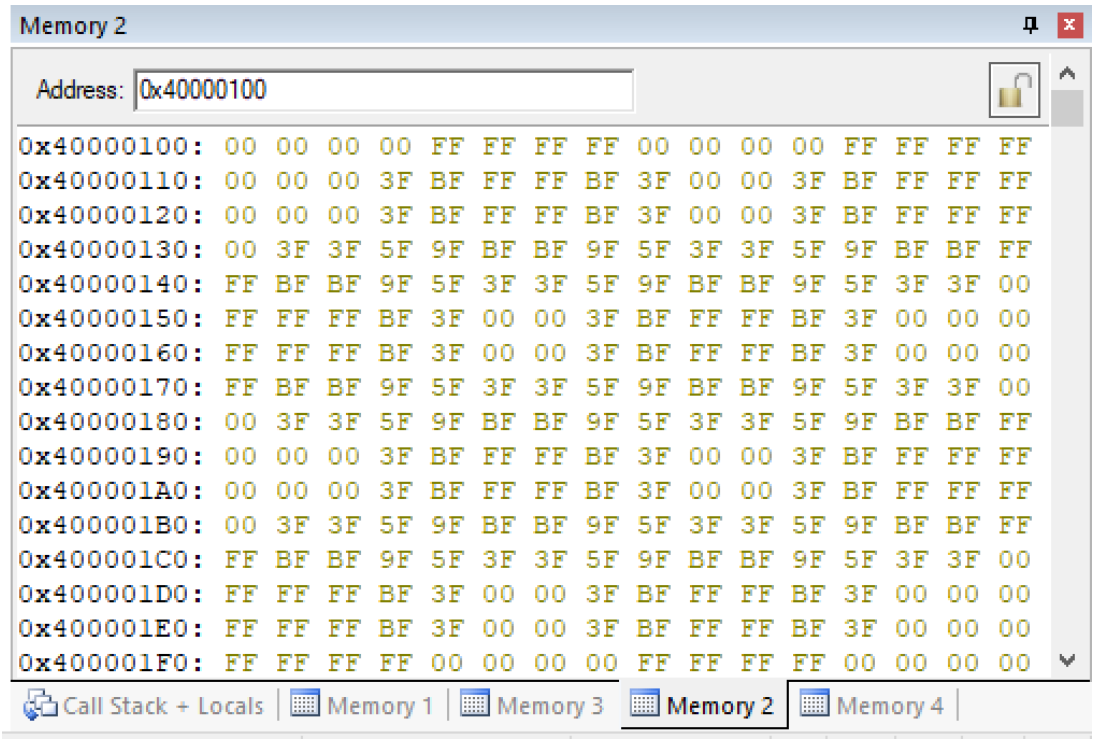


Figure 3.4 complted filtered image

Before image was filtered in address 0x40000000



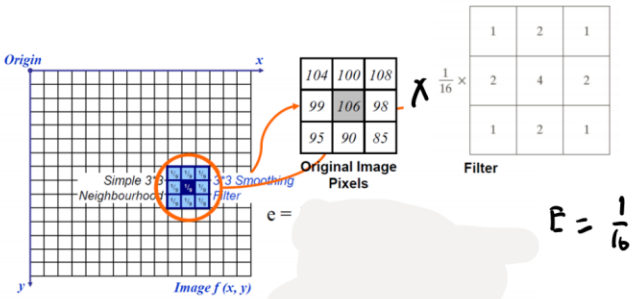
After image was filtered in address 0x40000100



Task 4

What did happen after applying the filter image? Explain your answer based on the new matrix values

Each filter image I calulating by using a 3x3 image of the orginal image the. The center of it is what being filtered in order to get it we multiply it by the corespong element of the kernel image which is [1,2,1:,2,4,2;,1,2,1]



$$E = \frac{1}{16}[(104\times1) + 2(100) + 1(108) + (2\times99) + (106\times4) + (98\times2) + (1\times95) + (90\times2) + (85\times1)]$$

$$= 99.375$$ ← fitted Pixel

Task 5

. Run your histogram program in part 2 one more time on the filtered image resulting from step 3 and obtain histogram data.

Once again I reused the exact same code I used to get the histogram for the unfiltered histogram only diffreenc is now we getting it for the filtered image the the type of intensity found in the filtered image will be stored in adress 0x40000500 and the number of pixel that have those corespsonding intensity will be stored in 0x40000300

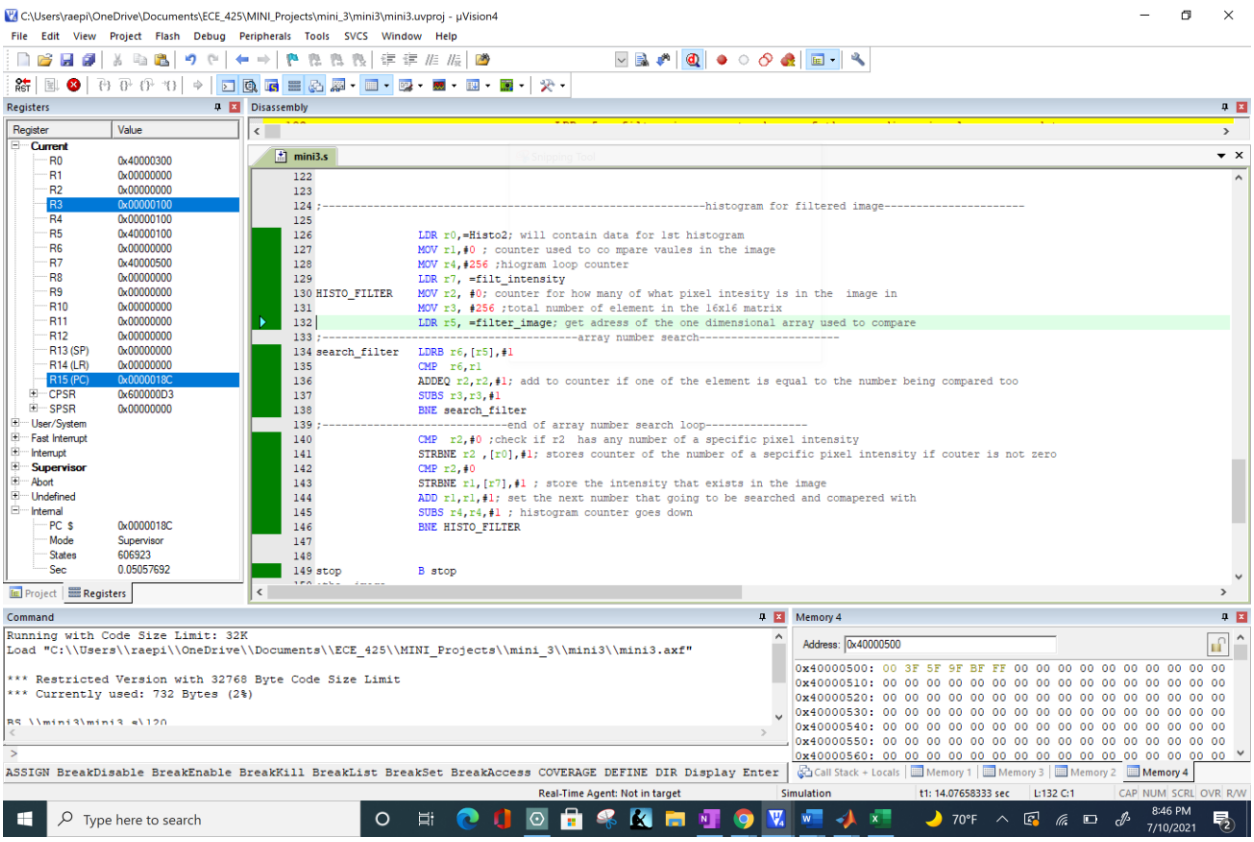


Figure 5.1 code for getting filtered histogram

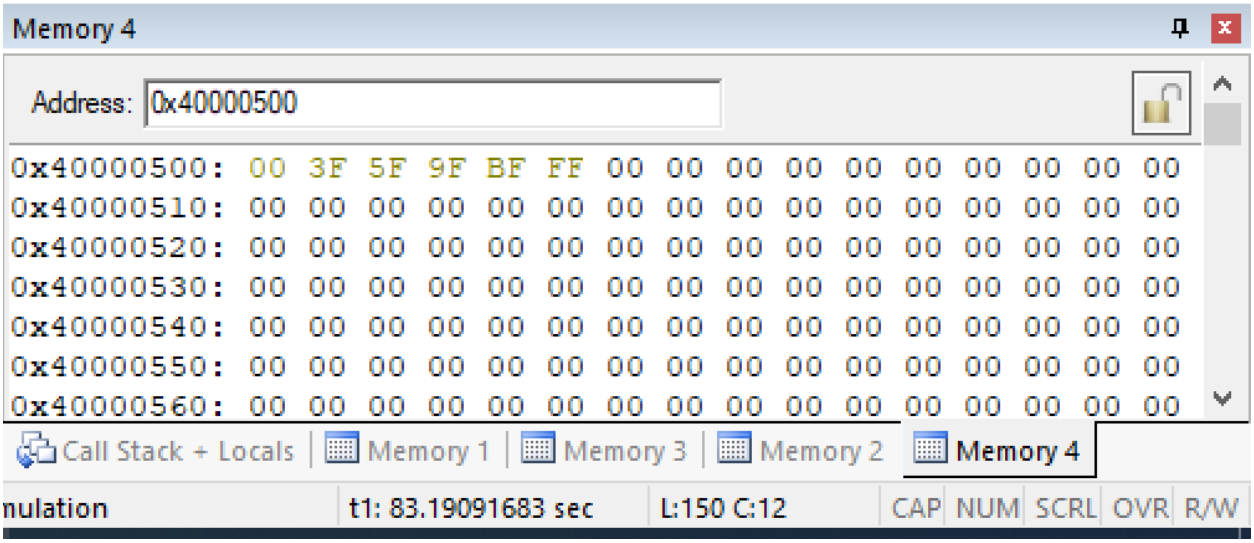


Figure 5.2 : 'Pixel intensity found in filterd image that bein g 0 , 63 (3F in hexa) , 95 (5F in hexa) , 159 (9F in hexa) , 191(BF in hexal) ,and 255 (FF in hexa)

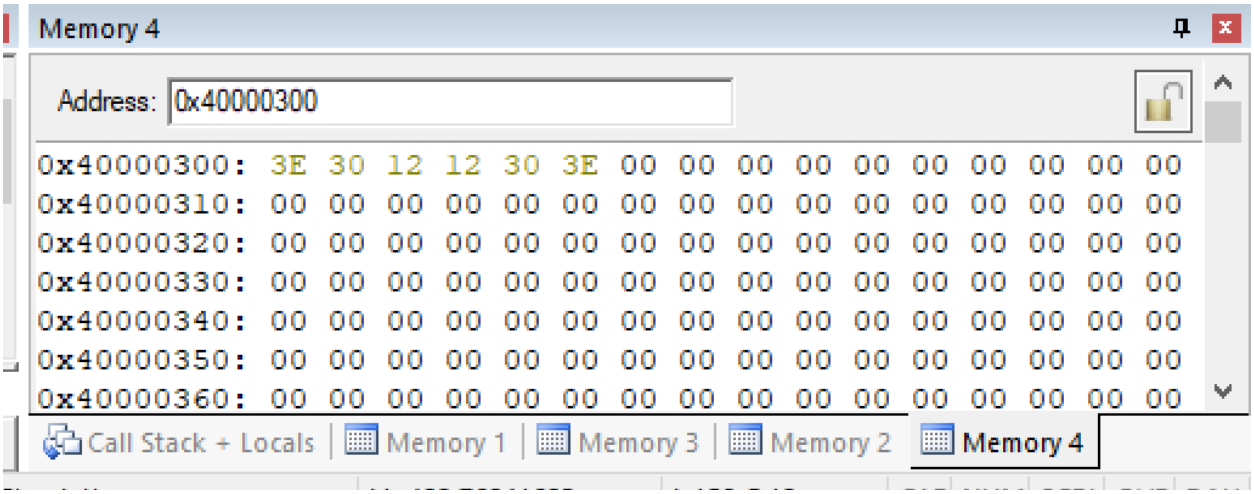
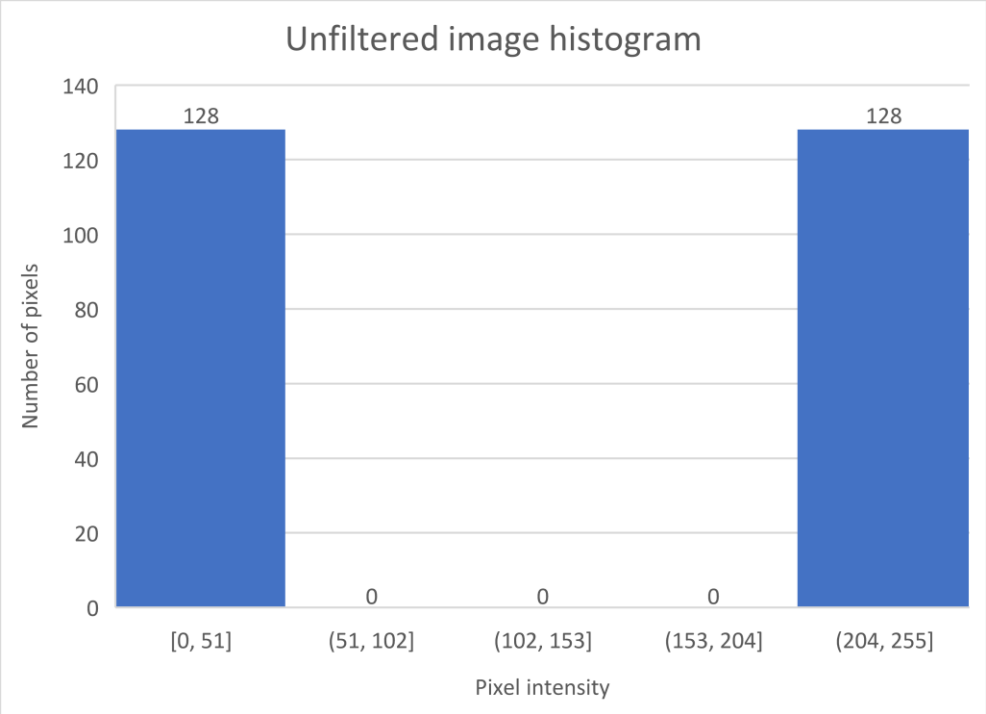


Figure 5.3 number pixel found with the found intensitys 62 pixel have intensity 0, 48 pixel have intensity 63 , 18 pixel have intensity 95, 18 pixel have intensity 151, 48 pixel have intensity 191 , and 62 pixel have intensity 255

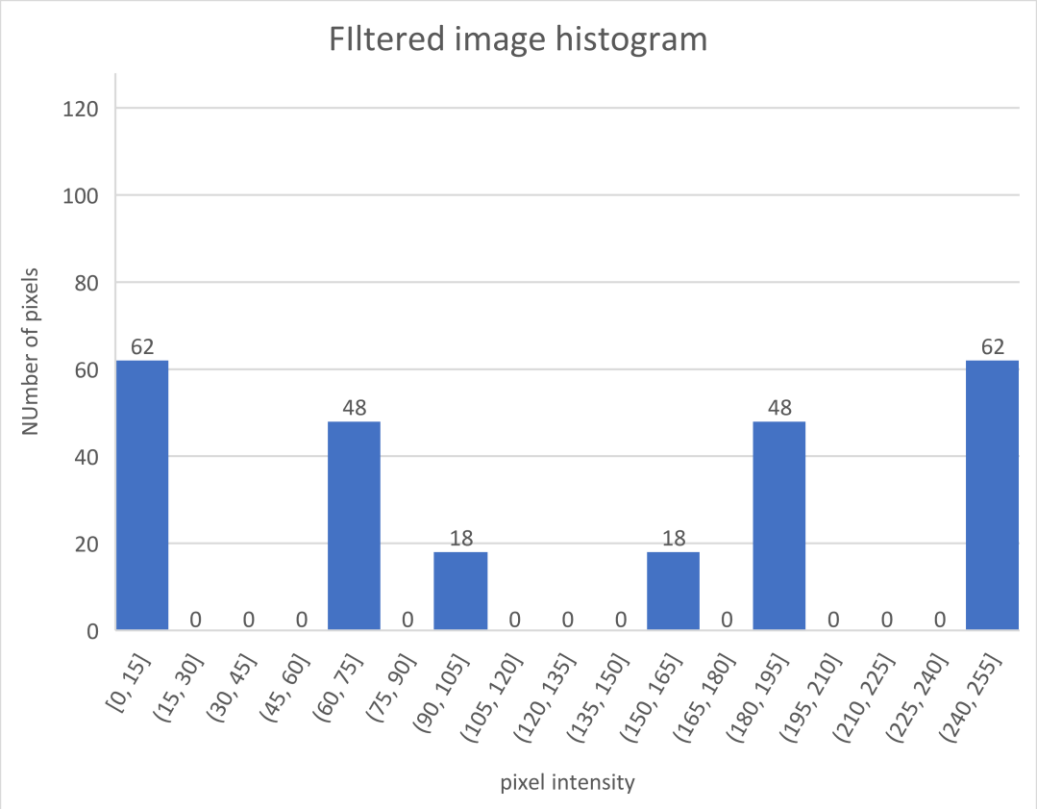
Task 6

Use Excel and data obtained in parts 2 and 5 to compare image histograms before and after applying the filter.

Unfiltered image histogram



Filtered image histogram



TASK 7

Conclusion

From doing this project I have significantly sharpened my programing skills. I was able to successfully complete all task that were asked in this project. I overall feel really accomplished for the handwork I put in to this program. The part I feel most proud of is the filter part which was defietly the most rigorous part of the project

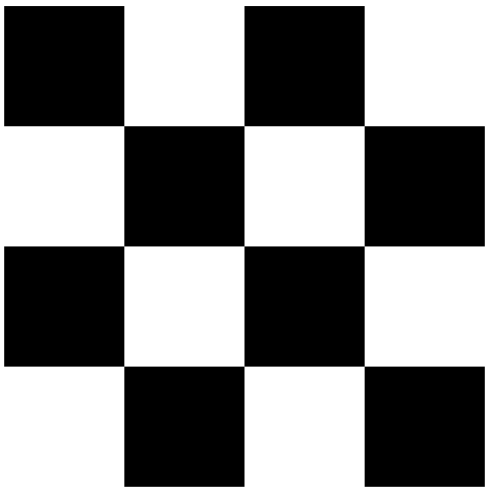
Task 8

.Use simple Matlab instructions to convert image data to images before and after filtering

Ryan mark

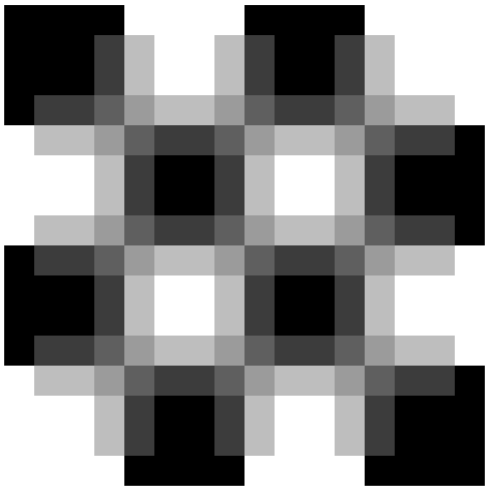
Tas 8 for mini project 3

```
IMAGE= [0,0,0,0,255,255,255,255,0,0,0,0,255,255,255,255;...
        0,0,0,0,255,255,255,255,0,0,0,0,255,255,255,255; ...
        0,0,0,0,255,255,255,255,0,0,0,0,255,255,255,255; ...
        0,0,0,0,255,255,255,255,0,0,0,0,255,255,255,255; ...
        255,255,255,255,0,0,0,0,255,255,255,255 ,0,0,0,0; ...
       255,255,255,255,0,0,0,0,255,255,255,255 ,0,0,0,0; ...
       255,255,255,255,0,0,0,0,255,255,255,255 ,0,0,0,0; ...
       255,255,255,255,0,0,0,0,255,255,255,255 ,0,0,0,0; ...
       0,0,0,0,255,255,255,255,0,0,0,0,255,255,255,255;...
        0,0,0,0,255,255,255,255,0,0,0,0,255,255,255,255; ...
        0,0,0,0,255,255,255,255,0,0,0,0,255,255,255,255; ...
        0,0,0,0,255,255,255,255,0,0,0,0,255,255,255,255; ...
        255,255,255,255,0,0,0,0,255,255,255,255 ,0,0,0,0; ...
       255,255,255,255,0,0,0,0,255,255,255,255 ,0,0,0,0; ...
       255,255,255,255,0,0,0,0,255,255,255,255 ,0,0,0,0; ...
       255,255,255,255,0,0,0,0,255,255,255,255 ,0,0,0,0];
  imn =uint8(IMAGE);
  imshow(imn,'InitialMagnification',3000)
```

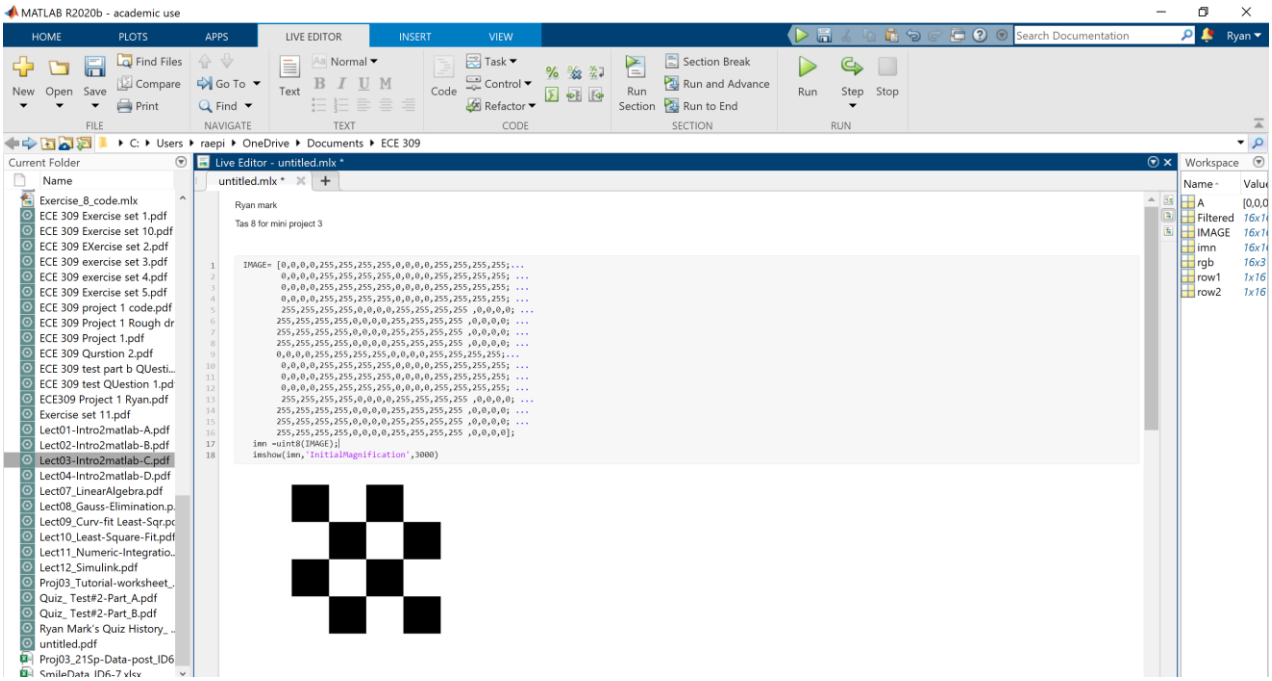


```
  Filtered =  [0,0,0,0,255,255,255,255,0,0,0,0,255,255,255,255;...
               0,0,0,63,191,255,255,191,63,0,0,63,191,255,255,255;...
               0,0,0,63,191,255,255,191,63,0,0,63,191,255,255,255;...
               0,63,63,95,159,191,191,159,95,63,63,95,159,191,191,255; ...
               255,191,191,159,95,63,63,95,159,191,191,159,95,63,63,0; ...
               255,255,255,191,63,0,0,63,191,255,255,191,63,0,0,0; ...
                255,255,255,191,63,0,0,63,191,255,255,191,63,0,0,0; ...
                255,191,191,159,95,63,63,95,159,191,191,159,95,63,63,0; ...
                0,63,63,95,159,191,191,159,95,63,63,95,159,191,191,255; ...
```

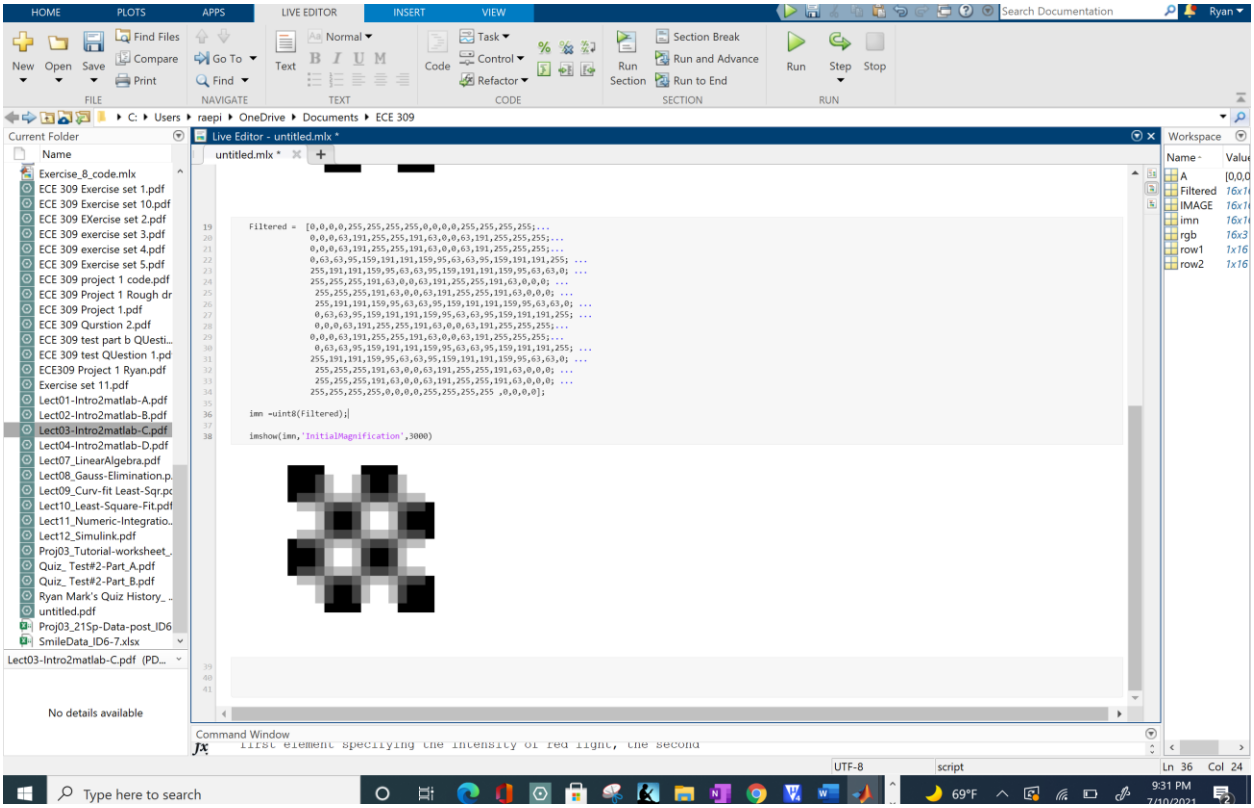
```
                0,0,0,63,191,255,255,191,63,0,0,63,191,255,255,255;...
               0,0,0,63,191,255,255,191,63,0,0,63,191,255,255,255;...
                0,63,63,95,159,191,191,159,95,63,63,95,159,191,191,255; ...
               255,191,191,159,95,63,63,95,159,191,191,159,95,63,63,0; ...
                255,255,255,191,63,0,0,63,191,255,255,191,63,0,0,0; ...
                255,255,255,191,63,0,0,63,191,255,255,191,63,0,0,0; ...
               255,255,255,255,0,0,0,0,255,255,255,255 ,0,0,0,0];

imn =uint8(Filtered);

imshow(imn,'InitialMagnification',3000)
```



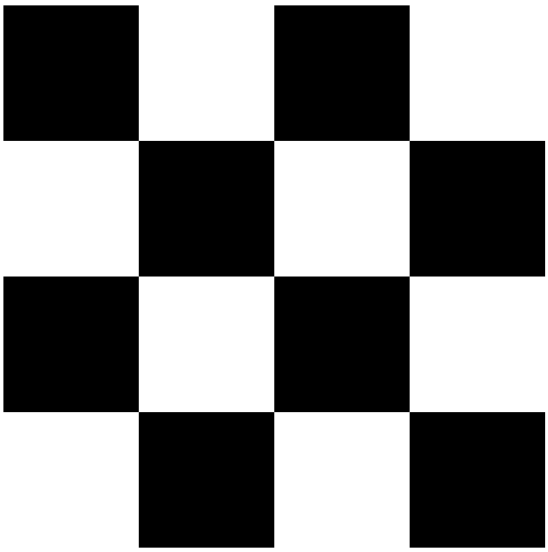
Unfiltered Image



Filtered Image



BEFORE



AFTER