مریم برازنده

۹۷۲۳۰۱۶

پروژه درس معماری کامپیوتر و ریزپردازنده

پاییز ۹۹

توضیح کلی الگوریتم

الگوریتم به این صورت خواهد بود که ابتدا آدرس اولین خانه سه خط اول اعداد تصویر را در رجیستر گذاشته ومحتوای آن ها را نیز ذخیره میکنیم .این کار را با جلو بردن رجیسترهای محتوی آدرس و دسترسی به سایر خانه ها ادامه میدهیم و به این ترتیب در هر بار گذشتن از LOOP ماتریس کرنل بر یک ماتریس سه در سه تصویر اعمال میشود. با استفاده از عملگرهای جمع و شیفت کرنل را اعمال میکنیم. مثلا برای ضرب کردن محتوای خانه ی اول در ۲, آن را یک واحد به سمت چپ شیفت میدهیم. برای ضرب کردن در -۴ آن را دو بار به چپ شیفت داده و از صفر کم میکنیم. سپس کرنل اعمال شده را در خانه 0x40010000 ممووری ذخیره کرده و بعدی ها را در ادامه این آدرس ذخیره میکنیم .این کار را با شمردن ۱۵ بار برای هر خط و ۱۵ بار برای هر ستون و رفتن به LOOP انجام میدهیم تا کرنل بر کل ماتریس اعمال شود. در ادامه با جزییات بیشتر توضیح داده می شود.

عملکرد هریک از بخش های کد

ابتدا داده های موجود در متلب قسمت noisy.txt برای کرنل گوسی و قسمت photo.txt را برای کرنل تشخیص لبه در آخر کد اضافه میکنیم تا دسترسی به تصویر داده باشیم. این کار را به صورت یک ماتریس دو بعدی انجام میدهیم و با توجه به اعداد برای بازدهی بیشتر از فرمت DCB استفاده میکنیم تا فضای کمتری اشغال کند و هر عدد به اندازه یک بایت ذخیره شود. حال در ابتدای این اعداد یک آدرس DCD تعریف کرده و label هایی که برای هر خط ماتریس زده بودیم را آنجا تعریف میکنیم. با این کار آدرس هر خط را ذخیره کرده ایم:

```
ROW   DCD ROW1,ROW2,ROW3,ROW4,ROW5,ROW6,ROW7,ROW8,ROW9,ROW10,ROW11,ROW12,ROW13,ROW14,ROW15,ROW16,ROW17
ROW1  DCB 129,129,109,153,143,118,158,144,42,102,175,157,133,114,177,72,72
ROW2  DCB 129,129,109,153,143,118,158,144,42,102,175,157,133,114,177,72,72
ROW3  DCB 102,102,110,157,109,97,111,114,6,102,99,86,122,122,183,151,151
ROW4  DCB 83,83,107,103,133,137,39,130,2,103,110,75,93,94,135,121,121
ROW5  DCB 105,105,99,144,81,116,80,125,48,102,107,108,77,95,100,108,108
ROW6  DCB 95,95,100,66,85,108,66,126,22,71,53,98,88,147,137,100,100
ROW7  DCB 192,192,73,79,119,119,136,113,7,112,85,80,141,132,36,87,87
ROW8  DCB 144,144,144,135,122,172,122,118,0,137,101,140,85,102,127,118,118
ROW9  DCB 32,32,28,27,0,25,0,29,42,38,14,0,34,0,0,59,59
ROW10 DCB 114,114,130,100,184,113,124,97,8,104,151,58,62,65,120,140,140
ROW11 DCB 122,122,44,116,78,82,141,93,0,111,57,63,99,61,110,139,139
ROW12 DCB 116,116,107,169,45,159,106,123,0,112,121,97,116,133,101,102,102
ROW13 DCB 68,68,40,158,88,100,143,115,57,141,153,114,48,62,117,81,81
ROW14 DCB 137,137,69,78,117,106,85,126,19,91,87,82,100,82,83,112,112
ROW15 DCB 145,145,144,132,95,121,148,85,67,72,166,153,87,80,77,127,127
ROW16 DCB 131,131,141,166,134,171,129,128,9,112,116,74,113,73,64,122,122
ROW17 DCB 131,131,141,166,134,171,129,128,9,112,116,74,113,73,64,122,122
```

شکل ۱ noisy.txt

```
RROW   DCD RROW1,RROW2,RROW3,RROW4,RROW5,RROW6,RROW7,RROW8,RROW9,RROW10,RROW11,RROW12,RROW13,RROW14,RROW15,RROW16,RROW17
RROW1  DCB 129,129,124,130,126,127,122,129,14,128,118,125,128,130,138,125,125
RROW2  DCB 129,129,124,130,126,127,122,129,14,128,118,125,128,130,138,125,125
RROW3  DCB 112,112,99,145,131,99,117,128,29,118,93,111,119,133,158,145,145
RROW4  DCB 105,105,97,104,111,134,96,127,11,125,114,98,109,129,114,129,129
RROW5  DCB 107,107,109,117,92,81,105,129,6,126,111,93,78,121,105,118,118
RROW6  DCB 101,101,99,75,101,100,108,122,0,125,76,79,90,94,122,118,118
RROW7  DCB 120,120,71,68,112,116,125,114,1,125,90,75,115,103,79,99,99
RROW8  DCB 129,129,126,127,126,130,128,118,2,115,115,111,119,129,127,128,128
RROW9  DCB 16,16,25,15,18,4,1,4,35,6,7,7,18,20,14,21,21
RROW10 DCB 128,128,126,128,128,127,115,118,8,125,120,87,90,108,96,122,122
RROW11 DCB 129,129,93,91,104,76,97,129,6,121,96,80,89,109,116,113,113
RROW12 DCB 129,129,117,102,91,108,90,128,14,115,108,111,105,90,109,100,100
RROW13 DCB 125,125,94,117,78,124,124,124,29,113,117,115,106,80,100,100,100
RROW14 DCB 120,120,95,81,119,87,103,127,31,109,111,111,87,86,86,114,114
RROW15 DCB 120,120,103,113,125,109,124,121,9,101,86,118,104,100,78,117,117
RROW16 DCB 128,128,128,130,145,127,123,123,0,114,95,93,112,84,105,122,122
RROW17 DCB 128,128,128,130,145,127,123,123,0,114,95,93,112,84,105,122,122
```

شکل ۲ photo.txt

برای اعمال کرنل آدرس سه خانه اول سه خط اول را به صورت زیر در سه رجیستر ذخیره میکنیم که محتوای آن به ما دسترسی به اعداد را میدهد:

```
MOV  R8, #15                ; register used for counting in rows
MOV  R9, #15                ; register used for counting in clomns
MOV32 R10, ROW      ; R0, R1, R2 points to the first array of first three rows
LDR  R0, [R10]
ADD  R10, R10, #4
LDR  R1, [R10]
ADD  R10, R10, #4
LDR  R2, [R10]
LDRB R3, [R0]
LDRB R4, [R1]
LDRB R5, [R2]
```

توجه میکنیم که چون آدرس DCD ذخیره شده یعنی محتوی یک WORD است پس برای دسترسی به آدرس خط بعدی باید ۴ تا اضافه کنیم.

رجیسترهای R8 , R9 برای شمارش ردیف و ستون به کار میروند.

حال وارد لوپ میشویم (شکل زیر):

```
LOOP1
        MOV  R6, #0                 ; R6 holds the sum of each kernel which is apllied
        LDRB R3, [R0], #1
        ADD  R6, R6, R3             ; start of applying [1 2 1] to the three arrays in row1 and so on
        LDRB R3, [R0], #1
        ADD  R6, R6, R3, LSL #1
        LDRB R3, [R0], #-1
        ADD  R6, R6, R3
        LDRB R4, [R1], #1
        ADD  R6, R6, R4, LSL #1     ; start of applying [2 4 2] to the three arrays in row2 and so on
        LDRB R4, [R1], #1
        ADD  R6, R6, R4, LSL #2
        LDRB R4, [R1], #-1
        ADD  R6, R6, R4, LSL #1
        LDRB R5, [R2], #1
        ADD  R6, R6, R5             ; start of applying [1 2 1] to the three arrays in row3 and so on
        LDRB R5, [R2], #1
        ADD  R6, R6, R5, LSL #1
        LDRB R5, [R2], #-1
        ADD  R6, R6, R5
        MOV  R6, R6, LSR #4
        STR  R6, [R11], #1
        SUBS R8, R8, #1             ; check if we finish one 3*1 rows or not (15 times)
        BNE  LOOP1                  ; if not start applying kernel on next arrays of same rows
        MOV  R8, #15
        ADD  R0, R0, #2             ; if it is finished so each row should be added 2 to skip the padding and edge
        ADD  R1, R1, #2
        ADD  R2, R2, #2
        SUBS R9, R9, #1             ; check if we,re not run out of number (15*15 times from the beginning)
        BNE  LOOP1                  ; if we,re not start over
```

رجیستر R6 کرنل اعمالی بر هر خانه را ذخیره میکند پس باید با هربار ورود به لوپ ست شود.

هر خانه ی حافظه که جمع میشود برای دسترسی به خانه ی بعدی رجیستر حاوی آدرس را یکی اضافه میکنیم. اما در هر لوپ هر رجیستر آدرس باید فقط یک خانه جلو رود تا هیچ خانه ای از ماتریس کل جا نیفتد.

خط ۹۲ رجیستر ۸ را یکی کم میکند تا وقتی مقدارش با شروع از ۱۵ به صفر رسید به خط بعدی ماتریس برود . این کار با کمک BNE انجام میشود که فلگ Z را چیک میکند (چون SUBS استفاده شده پس این فلگ به روز رسانی میشود).

برای رفتن به خط بعد باید دو واحد به هر رجیستر محتوی آدرس اضافه شود به خاطر حاشیه ای که برای PADDING درنظر گرفته شده.

بعد از انجام کرنل گوسی کرنل تشخیص لبه بر ماتریس خود اعمال میشود:

```
108             MOV  R9, #15                ; register used for counting in clomns
109             MOV  R12, #0
110             MOV32 R10,RROW
111             LDR  R0, [R10], #4              ; R0, R1, R2 points to the first array of first three rows
112             ADD  R0, R0, #1
113             LDR  R1, [R10], #4
114             LDR  R2, [R10]
115             ADD  R2, R2, #1
116
117 LOOP2       MOV  R6, #0
118             LDRB R3, [R0], #1           ; after we load r1, r0 increases by one
119             ADD  R6, R6, R3
120             LDRB R4, [R1], #1
121             ADD  R6, R6, R4
122             LDRB R4, [R1], #1
123             MOV  R4, R4, LSL #2
124             SUBS R4, R12, R4
125             ADD  R6, R6, R4
126             LDRB R4, [R1], #-1
127             ADD  R6, R6, R4
128             LDRB R5, [R2], #1
129             ADD  R6, R6, R5
130             MOV  R6, R6, LSR #2
131             ADD  R6, R6, #64
132             STR  R6, [R11], #1          ; this kernel stores after the first one
133             SUBS R8, R8, #1
134             BNE  LOOP2
135             MOV  R8, #15
136             ADD  R0, R0, #2             ; if it is finished so each row should be added 2 to skip the padding and edge
137             ADD  R1, R1, #2
138             ADD  R2, R2, #2
139             SUBS R9, R9, #1             ; check if we,re not run out of number (15*15 times from the beginning)
140             BNE  LOOP2
```

الگوریتم مانند حالت قبلیست و برای ضرب در -۴ ابتدا با شیفت ضرب در ۴ کرده سپس از ۰ کم میکنیم.

برای ذخیره کردن در حافظه به این صورت عمل میکنیم که یک آدرس دلخواه به رجیستری مثلا R11 داده شده و ذخیره میکنیم (خط ۱۳۲).

فعال سازی کلاک :

با استفاده از datasheet داریم:

آدرس بیس شروع RCC : 0x40023800

مقدار آفست : 0x30

پس آدرس رجیستر میشود :0x40023830

برای تنظیم پورت های : a,b

با استفاده از مقادیر موجود در دیتا شیت محاسبه میکنیم

- 0xA800 0000 for port A
- 0x0000 0280 for port B
- 0x0000 0000 for other ports

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
MODER15[1:0]		MODER14[1:0]		MODER13[1:0]		MODER12[1:0]		MODER11[1:0]		MODER10[1:0]		MODER9[1:0]		MODER8[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
MODER7[1:0]		MODER6[1:0]		MODER5[1:0]		MODER4[1:0]		MODER3[1:0]		MODER2[1:0]		MODER1[1:0]		MODER0[1:0]	
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 2y:2y+1 **MODERy[1:0]:** Port x configuration bits (y = 0..15)
These bits are written by software to configure the I/O direction mode.
00: Input (reset state)
01: General purpose output mode
10: Alternate function mode
11: Analog mode

برای تنظسم سرعت باید رجیستر GPIOB_OSPEEDRرا تغییر دهیم که خروجی اش یک شود.

سایر رجیستر ها هم به همین صورت.

Bits 15:0 **OTy[1:0]:** Port x configuration bits (y = 0..15)
These bits are written by software to configure the output type of the I/O port.
0: Output push-pull (reset state)
1: Output open-drain

Bits 2y:2y+1 **OSPEEDRy[1:0]:** Port x configuration bits (y = 0..15)
These bits are written by software to configure the I/O output speed.
00: 2 MHz Low speed
01: 25 MHz Medium speed
10: 50 MHz Fast speed
11: 100 MHz High speed on 30 pF (80 MHz Output max speed on 15 pF)

```
14  GPIOC_IDR       EQU     0x40011008
15  GPIOC_ODR       EQU     0x4001100C
16  GPIOC_BSRR      EQU     0x40011010
17  GPIOC_BRR       EQU     0x40011014
18  GPIOC_LCKR      EQU     0x40011018
19  RCC_AHB1ENR     EQU     0x40023830
20  GPIOA_MODER     EQU     0x40020000
21  GPIOB_MODER     EQU     0x40020400
22  GPIOA_OSPEEDR   EQU     0x40020008
23  GPIOB_OSPEEDR   EQU     0x40020408
24  GPIOA_PUPDR     EQU     0x4002000C
25  GPIOB_PUPDR     EQU     0x4002040C
26  GPIOB_OTYPER    EQU     0x40020404
27  ;*******************************************************
28                  ENTRY
29  Reset_Handler
30  ;*******************************************************
31  ; You'd want code here to enable GPIOC clock in AHB
32
33              MOV32 R0, #RCC_AHB1ENR
34              LDR   R1, [R0]
35              ORR   R1, R1, #3
36              STR   R1, [R0]
37
38              MOV32 R0, #GPIOA_MODER
39              LDR   R1, [R0]
40              AND   R1, R1, #0xFFFFFFFC
41              STR   R1, [R0]
42
43              MOV32 R0, #GPIOA_MODER
44              LDR   R1, [R0]
45              ORR   R1, R1, #4
46              STR   R1, [R0]
47
```

که در ابتدای کد انها را DEFINE کرده ایم و بعد تنظیم میکنیم

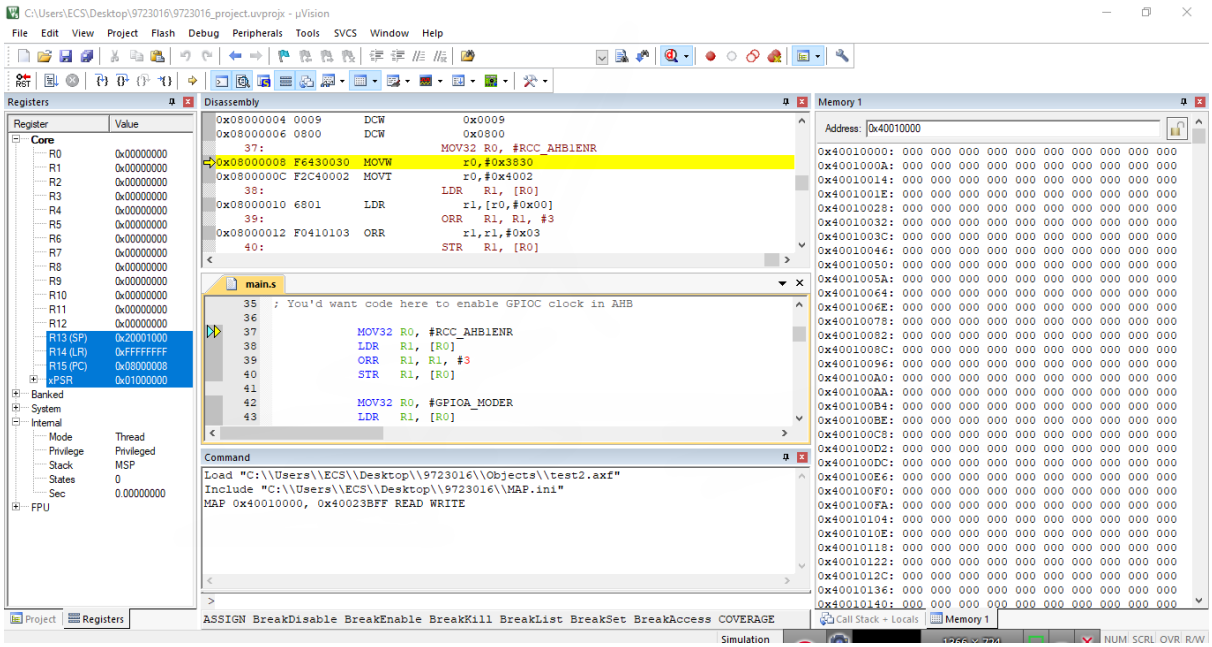
نتایج حاصل از اجرای کد

پس از این که کد را بیلد کردیم وارد بخش دیباگ میشویم و کد را اجرا میکنیم که حاصل احرا به شکل زیر است.

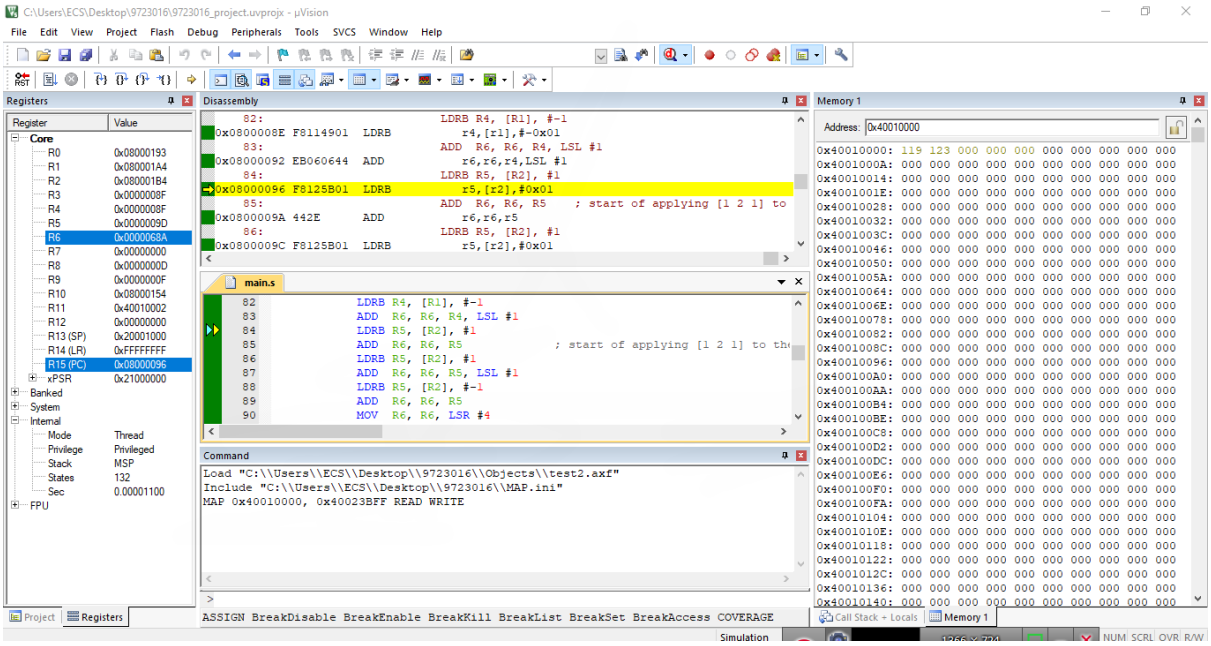
آدرسی که نتایج را در آن ذخیره کرده بودیم را در قسمت memory داده و مقادیر را بدست می آوریم .

در ابتدا مقادیر ست شذه اند و با F11 آن را جلو برده تا به لوپ بی نهایت رسیده و متوقف شویم

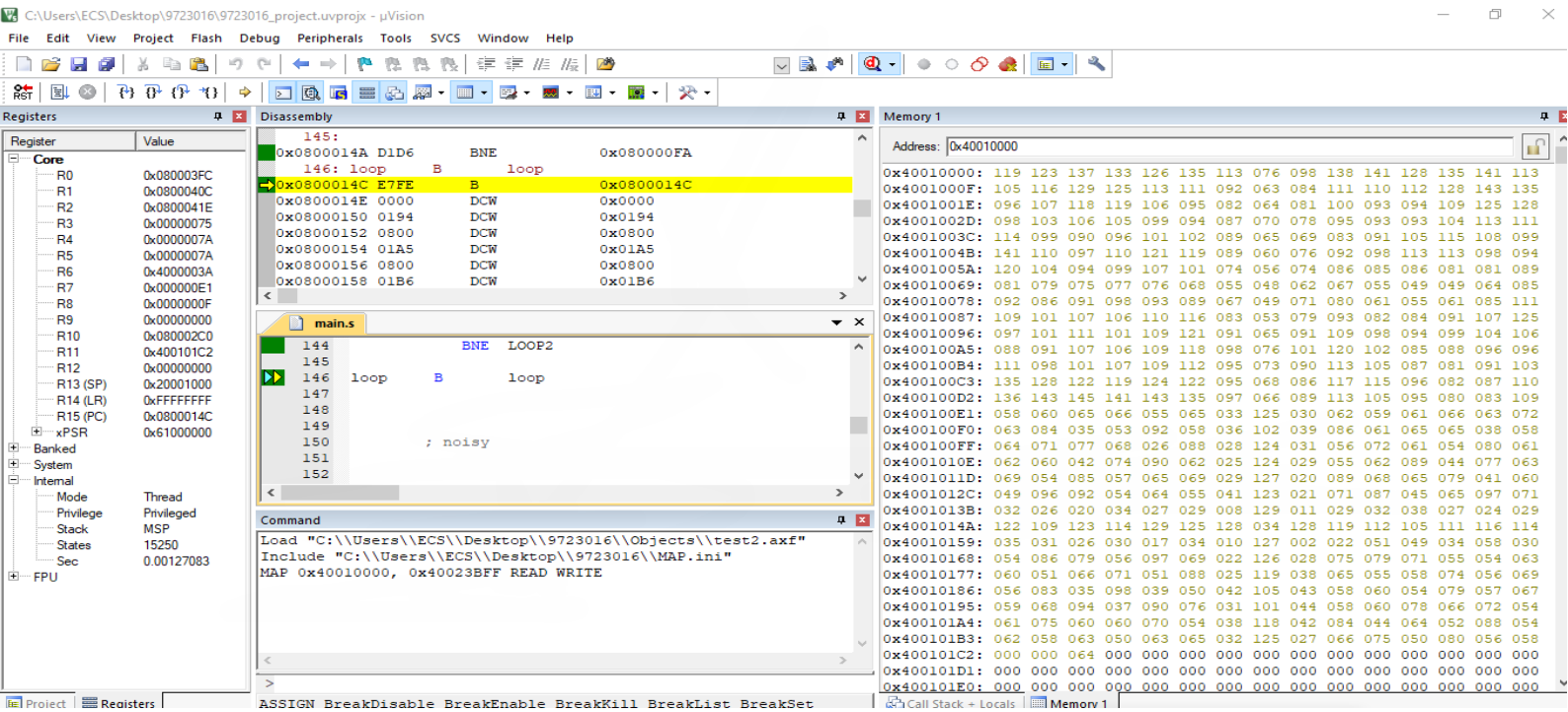
در ابتدا :



سپس با جلو رفتن مقادیر تغییر کرده و به مموری وارد میشوند

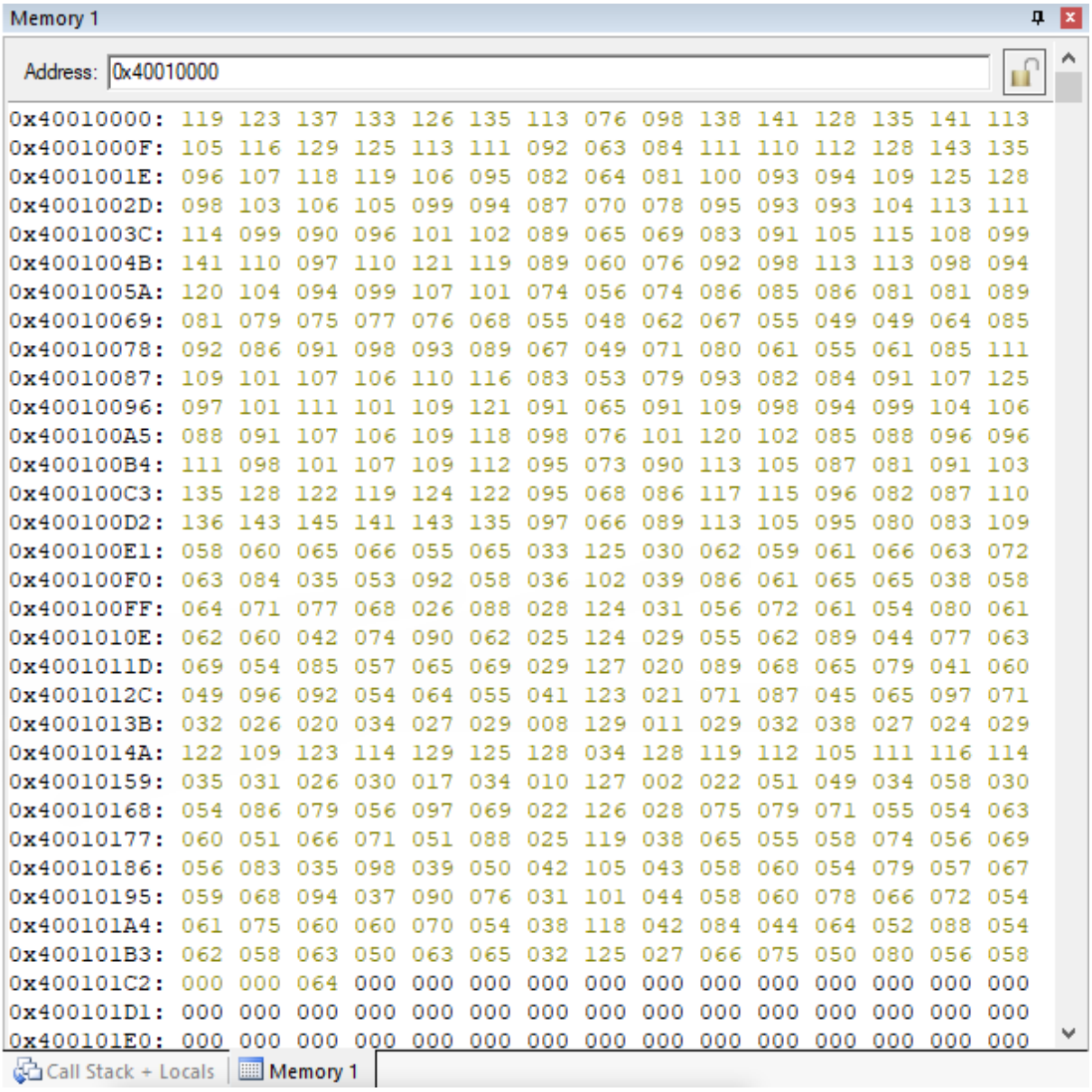


و در آخر:



نتایج نهایی:

۱۵ خط اول حاصل کرنل گوسی و



خط دوم حاصل کرنل لبه هستند .