بسمه تعالی

پروژه پایانی درس معماری کامپیوتر

عنوان پروژه: پردازش تصویر به زبان اسمبلی

نام استاد: سرکارخانم سیدین

نام و نام‌خانوادگی:

سجاد قدیری

محمد برآبادی

شماره دانشجویی:

۹۷۲۳۰۶۷

۹۷۲۳۰۱۵

پاییز ۱۴۰۰

توضیح کد:

کد دارای دو بخش کلی می باشد. بخش اول پروژه که شامل کرنل های گوسی هموار ساز و تشخیص لبه می باشد و بخش دوم آن که مقدار دهی تنظیمات میکروکنترلر است.

- **بخش اول:**

در این بخش آرایه دو بعدی به نام های Image و NoiseyImage تعریف شده است که ردیف های آن ها به صورت dcd (۳۲ بیت) و ستون های آن ها به صورت dcb (۸ بیت) تعریف شده است؛این ماتریس ها ۱۷*۱۷ هستند که حاوی اطلاعات عکس اصلی padding شده و عکس نویزی padding شده می باشند. (که ما هم میتوانیم از zero padding هم از symmetric padding استفاده کنیم اما برای اینکه با وجود kernel ۳*۳ داده‌ای را از دست ندهیم از symmetric padding استفاده میکنیم)در انتهای کد هم یک ماتریس ۱۵*۱۵ به نام NewImage به همین شکل تعریف شده با این تفاوت که به صورت readwrite تعریف شده است. این ماتریس برای ذخیره سازی عکس بعد از اعمال فیلتر ها می باشد.

```
NoiseyImage dcd     rowN0,rowN1,rowN2,rowN3,rowN4,rowN5,rowN6,rowN7,rowN8,rowN9,rowN10,rowN11,rowN12,rowN13,rowN14,rowN15,rowN16
rowN0   dcb     129,129,109,153,143,118,158,144,42,102,175,157,133,114,177,72,72
rowN1   dcb     129,129,109,153,143,118,158,144,42,102,175,157,133,114,177,72,72
rowN2   dcb     102,102,110,157,109,97,111,114,6,102,99,86,122,122,183,151,151
rowN3   dcb     83,83,107,103,133,137,39,130,2,103,110,75,93,94,135,121,121
rowN4   dcb     105,105,99,144,81,116,80,125,48,102,107,108,77,95,100,108,108
rowN5   dcb     95,95,100,66,85,108,66,126,22,71,53,98,88,147,137,100,100
rowN6   dcb     192,192,73,79,119,119,136,113,7,112,85,80,141,132,36,87,87
rowN7   dcb     144,144,144,135,122,172,122,118,0,137,101,140,85,102,127,118,118
rowN8   dcb     32,32,28,27,0,25,0,29,42,38,14,0,34,0,0,59,59
rowN9   dcb     114,114,130,100,184,113,124,97,8,104,151,58,62,65,120,140,140
rowN10  dcb     122,122,44,116,78,82,141,93,0,111,57,63,99,61,110,139,139
rowN11  dcb     116,116,107,169,45,159,106,123,0,112,121,97,116,133,101,102,102
rowN12  dcb     68,68,40,158,88,100,143,115,57,141,153,114,48,62,117,81,81
rowN13  dcb     137,137,69,78,117,106,85,126,19,91,87,82,100,82,83,112,112
rowN14  dcb     145,145,144,132,95,121,148,85,67,72,166,153,87,80,77,127,127
rowN15  dcb     131,131,141,166,134,171,129,128,9,112,116,74,113,73,64,122,122
rowN16  dcb     131,131,141,166,134,171,129,128,9,112,116,74,113,73,64,122,122

Image   dcd     row0,row1,row2,row3,row4,row5,row6,row7,row8,row9,row10,row11,row12,row13,row14,row15,row16
row0    dcb     129,129,124,130,126,127,122,129,14,128,118,125,128,130,138,125,125
row1    dcb     129,129,124,130,126,127,122,129,14,128,118,125,128,130,138,125,125
row2    dcb     112,112,99,145,131,99,117,128,29,118,93,111,119,133,158,145,145
row3    dcb     105,105,97,104,111,134,96,127,11,125,114,98,109,129,114,129,129
row4    dcb     107,107,109,117,92,81,105,129,6,126,111,93,78,121,105,118,118
row5    dcb     101,101,99,75,101,100,108,122,0,125,76,79,90,94,122,118,118
row6    dcb     120,120,71,68,112,116,125,114,1,125,90,75,115,103,79,99,99
row7    dcb     129,129,126,127,126,130,128,118,2,115,115,111,119,129,127,128,128
row8    dcb     16,16,25,15,18,4,1,4,35,6,7,7,18,20,14,21,21
row9    dcb     128,128,126,128,128,127,115,118,8,125,120,87,90,108,96,122,122
row10   dcb     129,129,93,91,104,76,97,129,6,121,96,80,89,109,116,113,113
row11   dcb     129,129,117,102,91,108,90,128,14,115,108,111,105,90,109,100,100
row12   dcb     125,125,94,117,78,124,124,124,29,113,117,115,106,80,100,100,100
row13   dcb     120,120,95,81,119,87,103,127,31,109,111,111,87,86,86,114,114
row14   dcb     120,120,103,113,125,109,124,121,9,101,86,118,104,100,78,117,117
row15   dcb     128,128,128,130,145,127,123,123,0,114,95,93,112,84,105,122,122
row16   dcb     128,128,128,130,145,127,123,123,0,114,95,93,112,84,105,122,122
```

این اطلاعات عکس‌ها هم در متلب استخراج شده‌اند و ما در این قسمت استفاده میکنیم.

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	129	124	130	126	127	122	129	14	128	118	125	128	130	138	125
2	112	99	145	131	99	117	128	29	118	93	111	119	133	158	145
3	105	97	104	111	134	96	127	11	125	114	98	109	129	114	129
4	107	109	117	92	81	105	129	6	126	111	93	78	121	105	118
5	101	99	75	101	100	108	122	0	125	76	79	90	94	122	118
6	120	71	68	112	116	125	114	1	125	90	75	115	103	79	99
7	129	126	127	126	130	128	118	2	115	115	111	119	129	127	128
8	16	25	15	18	4	1	4	35	6	7	7	18	20	14	21
9	128	126	128	128	127	115	118	8	125	120	87	90	108	96	122
10	129	93	91	104	76	97	129	6	121	96	80	89	109	116	113
11	129	117	102	91	108	90	128	14	115	108	111	105	90	109	100
12	125	94	117	78	124	124	124	29	113	117	115	106	80	100	100
13	120	95	81	119	87	103	127	31	109	111	111	87	86	86	114
14	120	103	113	125	109	124	121	9	101	86	118	104	100	78	117
15	128	128	130	145	127	123	123	0	114	95	93	112	84	105	122

اطلاعات عکس اصلی در متلب

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	129	109	153	143	118	158	144	42	102	175	157	133	114	177	72
2	102	110	157	109	97	111	114	6	102	99	86	122	122	183	151
3	83	107	103	133	137	39	130	2	103	110	75	93	94	135	121
4	105	99	144	81	116	80	125	48	102	107	108	77	95	100	108
5	95	100	66	85	108	66	126	22	71	53	98	88	147	137	100
6	192	73	79	119	119	136	113	7	112	85	80	141	132	36	87
7	144	144	135	122	172	122	118	0	137	101	140	85	102	127	118
8	32	28	27	0	25	0	29	42	38	14	0	34	0	0	59
9	114	130	100	184	113	124	97	8	104	151	58	62	65	120	140
10	122	44	116	78	82	141	93	0	111	57	63	99	61	110	139
11	116	107	169	45	159	106	123	0	112	121	97	116	133	101	102
12	68	40	158	88	100	143	115	57	141	153	114	48	62	117	81
13	137	69	78	117	106	85	126	19	91	87	82	100	82	83	112
14	145	144	132	95	121	148	85	67	72	166	153	87	80	77	127
15	131	141	166	134	171	129	128	9	112	116	74	113	73	64	122

اطلاعات عکس نویزی در متلب

در ابتدای برنامه کاربر میتواند تعیین کند که از کدام یک از دو فیلتر موجود استفاده کند و برای همین از یک تابع به نام IsGaussionKernel استفاده کردیم.

```
;decide between gaussion filter and edge detection filter
MOV32   r10, #IsGaussionKernel
LDR     r10, [r10]
CMP     r10, #0
BNE     GaussionStart   ;skip edge detection filter
```

در ادامه هم دو تابع برای فیلترها داریم:

تابع GaussianKernel برای اعمال فیلتر گوسی به عکس نویزی می باشد. باید توجه کرد که چون در این تابع توابع GetIndexN و SetIndex صدا زده می شوند و تعریف تابع باید به گونه ای باشد که بعد از اجرای آن، دستوری که بعد از دستور صدا زدن این تابع قرار دارد، اجرا شود (یعنی pc مقدار قبلی خود را بگیرد.) در ابتدای این تابع باید مقدار link register را در یک رجیستر ذخیره کرد که در انتهای تابع مقدار رجیستر در pc ریخته شود.

در این تابع رجیستر های r3 و r4 وردی می باشند که به ترتیب سطر و ستون درایه مورد نظر برای اعمال فیلتر گوسی را مشخص می کنند. این تابع درایه مورد نظر را درایه وسط یک ماتریس ۳*۳ در نظر گرفته و مقادیر آن را نظیر به نظیر در ماتریس کرنل گوسی ظرب می کند و همه ی این مقادیر را جمع کرده، بر ۱۶ تقسیم میکند و در درایه ای از ماتریس NewImage می ریزد که که هر دو عدد سطر و ستون آن یک واحد از r3 وr4 کمتر باشند.

EdgeDetectKernel هم مانند GaussianKernel یک تابع با ورودی های r3 و r4 می باشد که درایه ورودی را درایه مرکزی یک ماتریس سه در سه در نظر می گیرد و درایه های ماتریس تشخیص لبه را نظیر به نظیر در آن ضرب می کند. برای ضرب کردن عدد منهای چهار در درایه وسط میتوان با استفاده از ۲ تا شیفت چپ آن را در ۴ ضرب کرد و بعد با دستور SBCS مقدار منفی آن را با مقدار رجیستری که حاوی جمع مقادیر Index های قبلی می باشد جمع کرد.

برای اینکه هریک از این کرنل های گوسی و تشخیص لبه روی تمام پیکسل های عکس اعمال شود برای هر یک از آن ها یک for تو در تو در نظر گرفته شده است. r8 و r9 به عنوان شمارنده های این for های تو درتو از پیکسل ۱ تا ۱۵ روی Image حرکت می کنند و به این ترتیب کرنل روی تمام پیکسل های عکس اعمال می شود.

```
            ;edge detection filter algorithem
            MOV32   r8, #1
            MOV32   r9, #1
MyLoop1     MOV     r3, r8
            MOV     r4, r9
            BL      EdgeDetectKernel
            ADD     r9, r9, #1
            CMP     r9, #16
            BNE     MyLoop1
            ADD     r8, r8, #1
            MOV32   r9,#1
            CMP     r8, #16
            BNE     MyLoop1
            B       loop    ;skip gaussion filter

            ;gaussion filter algorithem
GaussionStart
            MOV32   r8, #1
            MOV32   r9, #1
MyLoop2     MOV     r3, r8
            MOV     r4, r9
            BL      GaussianKernel
            ADD     r9, r9, #1
            CMP     r9, #16
            BNE     MyLoop2
            ADD     r8, r8, #1
            MOV32   r9,#1
            CMP     r8, #16
            BNE     MyLoop2
```

و برای پیاده‌سازی بهتر و بهینه‌تر سه تابع دیگر استفاده میکنیم تا به اطلاعات عکس‌های اصلی و نویزی دسترسی سریعتری داشته باشیم و همچنین به سادگی عکس‌های فیلتر شده جدید را بسازیم.

```
SetIndex    MOV32   r6, #4
            MUL     r0, r0,r6
            ;MUL    r1, r1,r6
            MOV32   r6, #NewImage
            LDR     r6, [r6,r0]
            STRB    r2, [r6,r1]
            MOV     pc, lr

GetIndex    MOV32   r2, #4
            MUL     r0, r0,r2
            ;MUL    r1, r1,r2
            MOV32   r2, #Image
            LDR     r2, [r2,r0]
            LDRB    r2, [r2,r1]
            MOV     pc, lr

GetIndexN   MOV32   r2, #4
            MUL     r0, r0,r2
            ;MUL    r1, r1,r2
            MOV32   r2, #NoiseyImage
            LDR     r2, [r2,r0]
            LDRB    r2, [r2,r1]
            MOV     pc, lr
```

تابع GetIndex برای دسترسی به هر درایه از ماتریس Image می باشد به این صورت که ردیف و ستون را به عنوان ورودی دریافت میکند (چون ردیف ها به صورت dcd تعریف شده ردیف را در ۴ ضرب میکند)

و در خروجی محتوای درایه مورد نظر را در رجیستر r2 می دهد.

تابع GetIndexN هم دقیقا عملکردی مشابه تابع GetIndex دارد با این تفاوت که امکان دسترسی به هر درایه از ماتریس NoiseyImage را فراهم می کند.

تابع SetIndex برای مقدار دهی در ماتریس نهاییِ NewImage تعریف شده است. در ورودی r0، r1 و r2 را دریافت میکند که به ترتیب ردیف و ستون و مقداری که باید در آن خانه از ماتریس قرار داده شود هستند.

این قسمت از کد هم وقتی در حلقه به خانه جدیدی میرسد فراخوانی میشود تا کانولوشن بین کرنل و یک ماتریس ۳*۳ دور خانه موردنظر اتفاق بیفتد که در اینجا الگوریتم آن مشاهده میشود.

```
GaussianKernel  MOV     r7, lr
                ADDS    r3, r3, #0      ;to clear carry bit
                SUB     r0, r3, #1
                SUB     r1, r4, #1
                BL      GetIndexN
                MOV     r5, r2
                SUB     r0, r3, #1
                MOV     r1, r4
                BL      GetIndexN
                MOV     r2, r2, LSL #1  ;r2 = r2*2
                ADCS    r5, r5, r2
                SUB     r0, r3, #1
                ADD     r1, r4, #1
                BL      GetIndexN
                ADCS    r5, r5, r2
                MOV     r0, r3
                SUB     r1, r4, #1
                BL      GetIndexN
                MOV     r2, r2, LSL #1  ;r2 = r2*2
                ADCS    r5, r5, r2
                MOV     r0, r3
                MOV     r1, r4
                BL      GetIndexN
                MOV     r2, r2, LSL #2  ;r2 = r2*4
                ADCS    r5, r5, r2
                MOV     r0, r3
                ADD     r1, r4, #1
                BL      GetIndexN
                MOV     r2, r2, LSL #1  ;r2 = r2*2
                ADCS    r5, r5, r2
                ADD     r0, r3, #1
                SUB     r1, r4, #1
                BL      GetIndexN
                ADCS    r5, r5, r2
                ADD     r0, r3, #1
                MOV     r1, r4
                BL      GetIndexN
                MOV     r2, r2, LSL #1  ;r2 = r2*2
                ADCS    r5, r5, r2
                ADD     r0, r3, #1
                ADD     r1, r4, #1
                BL      GetIndexN
                ADCS    r5, r5, r2
                MOV     r5, r5, LSR #4  ;r5=r5/16
                SUB     r0, r3, #1
                SUB     r1, r4, #1
                MOV     r2, r5
                BL      SetIndex
                MOV     pc, r7
```

1	2	1
2	4	2
1	2	1

$\times \frac{1}{16}$

(آ) کرنل گوسی هموارساز

و به همین طریق برای فیلتر تشخیص لبه:

```
EdgeDetectKernel    MOV     r7, lr
                    ADDS    r3, r3, #0  ;to clear carry bit
                    SUB     r0, r3, #1
                    MOV     r1, r4
                    BL      GetIndex
                    MOV     r5, r2

                    MOV     r0, r3
                    SUB     r1, r4, #1
                    BL      GetIndex
                    ADCS    r5, r5, r2

                    MOV     r0, r3
                    MOV     r1, r4
                    BL      GetIndex
                    MOV     r2, r2, LSL #2  ;r2 = r2*4
                    SBCS    r5, r5, r2      ;r5 = r5-r2

                    MOV     r0, r3
                    ADD     r1, r4, #1
                    BL      GetIndex
                    ADCS    r5, r5, r2

                    ADD     r0, r3, #1
                    MOV     r1, r4
                    BL      GetIndex
                    ADCS    r5, r5, r2

                    MOV     r5, r5, ASR #2  ;r5=r5/4
                    ADCS    r5, r5, #64
                    SUB     r0, r3, #1
                    SUB     r1, r4, #1
                    CMP     r5, #100
                    MOVLS       r5, #0
                    MOVHI       r5, #255
                    MOV     r2, r5
                    BL      SetIndex
                    MOV     pc, r7
```

0	1	0
1	-4	1
0	1	0

(ب) کرنل تشخیص لبه

- **بخش دوم:** کد با توجه به موارد خواسته شده رجیستر های مورد نظر را تعریف شده اند و از صفحه ی ۵۰ دیتا شیت آدرس شروع آن ها برداشته شده و به آن آدرسی مقدار offset (که آن هم برای هر رجستر در دیتا شیت ذکر شده) مربوطه اضافه شده است.

```
RCC_AHB1ENR     EQU     0x40023830
GPIOA_MODER     EQU     0x40020000
GPIOB_MODER     EQU     0x40020400
GPIOB_OTYPER    EQU     0x40020404
GPIOA_OSPEEDR   EQU     0x40020008
GPIOB_OSPEEDR   EQU     0x40020408
GPIOA_PUPDR     EQU     0x4002000C
GPIOB_PUPDR     EQU     0x4002040C
GPIOA_IDR       EQU     0x40020010
GPIOB_ODR       EQU     0x40020414
```

رجیستر RCC_AHB1ENR برای فعال سازی کلاک پورت های A و Bمی‌باشد،رجیسترهای GPIOA_MODE و GPIOB_MODER برای تعیین مد ورودی و خروجی pinA0 و pinB1،رجیستر GPIOB_OTYPER برای تعیین pinB1 از نوع push pull، رجیسترهای GPIOA_OSPEEDR و GPIOB_OSPEEDR برای تعیین سرعت پین‌های مذکور،رجیسترهای GPIOA_PUPDR و GPIOB_PUPDR هم برای تعیین pull up/down می باشند.

برای مقدار دهی به این رجیستر ها باید دقت کرد که این رجیستر ها ۳۲ بیتی هستند و ما فقط یک بیت یا دو بیت آن ها را (بسته به نوع تعریف آن ها در دیتا شیت که به هر پین یک بیت اختصاص داده شده یا دو پین) میخواهیم مقدار مشخصی در آن بریزیم و نمی خواهیم مقادیر بقیه بیت ها تغییری کند به همین دلیل همان طور که در توضیحات پروژه آمده از AND و OR استفاده می کنیم.(اگر نیاز به صفر کردن بیتی هستیم آن بیت را با صفر AND و اگر نیاز به یک کردن بیتی داریم آن بیت را با یک OR میکنیم)

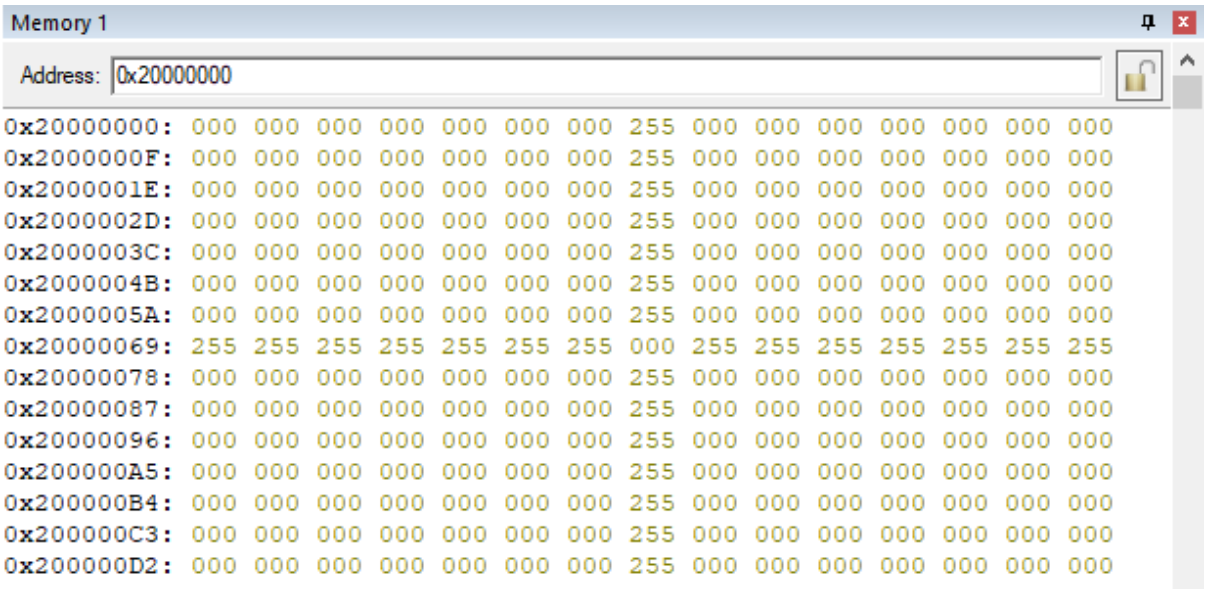
برای این قسمت هم میتوان از صفحه ۱۱۰ به بعد و ۱۴۸ به بعد استفاده کنیم.

نتایج نهایی :

در نهایت با اعمال دو فیلتر نتایج بدست آمده را با نتایج متلب مقایسه میکنیم تا از صحت و دقت الگوریتم و برنامه خود مطلع شویم.

کرنل تشخیص لبه:

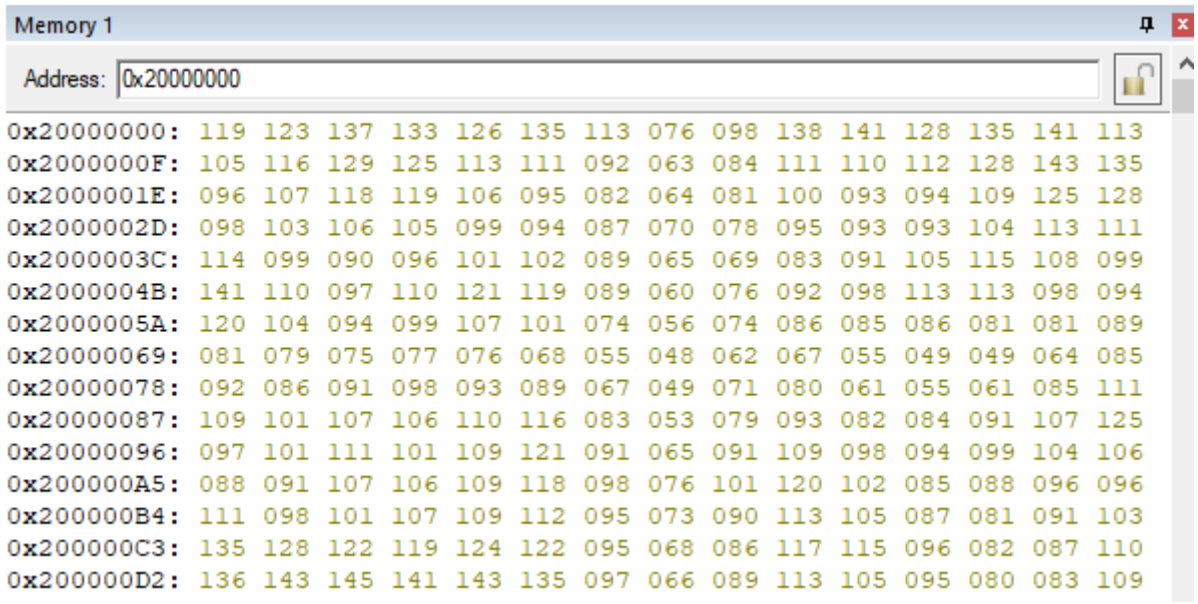
	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	0	0	0	0	0	0	0	255	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	255	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	255	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	255	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	255	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	255	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	255	0	0	0	0	0	0	0
8	255	255	255	255	255	255	255	0	255	255	255	255	255	255	255
9	0	0	0	0	0	0	0	255	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	255	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	255	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	255	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	255	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	255	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	255	0	0	0	0	0	0	0



همانطور که مشاهده میکنیم کاملا دو جواب یکسان هستند و این نشان‌دهنده صحت و درستی ۱۰۰درصدی در روند طراحی فیلتر دارد.

کرنل گوسی هموارساز:

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	119	124	138	134	127	135	113	76	98	138	141	129	135	141	113
2	105	116	129	125	113	112	92	64	85	111	110	112	128	144	135
3	96	108	118	119	106	95	83	64	82	100	94	95	109	126	129
4	98	103	106	106	100	95	87	70	78	95	93	93	104	113	111
5	115	99	91	97	101	102	89	65	69	84	91	106	115	109	100
6	141	110	97	110	121	119	90	61	77	92	98	114	114	99	95
7	120	104	94	99	107	102	74	56	75	87	85	87	82	81	90
8	81	79	76	77	76	69	55	48	63	67	56	50	50	65	86
9	92	87	92	98	94	89	67	49	72	80	62	55	62	86	112
10	109	102	107	107	111	117	83	54	79	93	83	85	91	108	125
11	98	102	111	102	110	122	91	65	91	109	98	95	99	104	106
12	89	92	108	107	109	119	98	77	101	120	103	86	88	97	97
13	111	99	102	107	109	113	95	73	90	113	106	88	82	91	104
14	136	129	122	119	124	123	95	68	87	117	115	97	82	88	110
15	136	144	145	141	144	136	98	67	89	113	106	95	81	83	109



مانند قسمت قبل تا ۹۹درصد اطلاعات درست میباشند و بعضی اعداد کمی با هم فرق دارند و آن هم خطای محاسباتی میباشد.