

Proyecto de Programación en Ensamblador

Curso 2018/2019

Memoria de proyecto
ensamblador 88110

Miguel Ángel Sastre Gálvez

Y160374

Contenido

- 1. Histórico del desarrollo de las rutinas3
- 2. Pruebas 34
- 3. Observaciones finales45
- 4. Anexo46
 - 4.1 Pruebas a mano46

1. Histórico del desarrollo de las rutinas

En este apartado se describe el desarrollo de cada subrutina, las fechas en las que se realizó y con una aproximación del tiempo que se invirtió el día indicado. Además, se va adjuntar el código realizado en papel como documentación de cómo se planteó inicialmente cada subrutina.

19/10/2018 – 3 horas

Inicio del proyecto. Creación del archivo de autores.txt, lectura del documento del proyecto y el subrayado de los puntos importantes de cara al 1º hito.


En este mismo día se realizó la subrutina “nFiltrados” en papel. En este punto, todavía había muchas dudas de como funcionaban realmente ciertas instrucciones y como recuperar valores de una dirección. Además, de la falta de soltura y confianza en cuanto al funcionamiento de la programación en ensamblador.

El código escrito no funciono demasiado bien al pasarlo al pc, hubo que hacer mas adelante mucho reajuste y con ello, se aprendieron nuevas cosas. Inicialmente, las 3 primeras subrutinas necesarias para el 1º hito no se tuvieron en cuenta guardar la dirección de retorno(r1) en la pila, pero más adelante se incluiría.

NTillados = ntillados (opu)

og 0
nt: 0

1. Hay que asegurarse que el log terminado con el valor.
2. Si se carga o actualiza NT al tiempo con ello.
3. Si se devuelve bien.

ntillados: ll r2, r30, r30; r2 = opu.  ¿Se está cargando el valor en r2?

~~or r2, r2, r0;~~

lfa (r2, NF); r3 = valor de NF. ¿Se está cargando el valor en r3?

Esto incrementa el valor de NF al valor de opu

Si $\left\{ \begin{array}{l} \text{cmp r7, r2, r0; Si opu = 0.} \\ \text{bbl ge, r7, ini; Si opu >= 0, se incrementa.} \\ \text{subu r3, r3, 1; Decrementar NF en 1.} \end{array} \right.$ Decrementar NF.

Si $\left\{ \begin{array}{l} \text{cmp r7, r3, r0;} \\ \text{bbl lt, r7, nFalero; Si r3 < 0, entonces nt = 0.} \end{array} \right.$

nt a menor de 0.

nFalero: ll r2, r0, r0; \rightarrow r2 se carga con 0 si AF es menor de 0.

Se carga en memoria.

$\left\{ \begin{array}{l} \text{ini: st r2, r30+0; Se carga} \\ \text{st r2, r29, 0;} \end{array} \right.$

jmp(r29); \rightarrow Se devuelve el valor de NF.

Se devuelve.

21/10/2018 y 23/10/2018 – 2-3 horas cada día

En este día se escribió a papel las subrutinas ActualizaFiltro y Comp.

Con lo aprendido con nFiltrados, la subrutina ActualizaFiltro no fue un problema construirla. De cara al código final con el que se ha implementado esta subrutina no hay demasiadas diferencias, aun así, a la hora de la entrega para el 1º hito tuvo errores debido a que no se tuvo en cuenta de que las operaciones debían realizarse con muls y divs, es decir, con signo. Un error simple y que pude arreglar rápidamente para cumplir con el 1º hito.

La subrutina Comp fue algo mas complicada al inicio, debido al desconocimiento de como venían las matrices y como se debían leer de la memoria. Al principio pensé que debía usarse alguna instrucción como extu o clr, hasta que caí en la cuenta de un ejercicio realizado en clase en el que se leían carácter por carácter y solo había que usar "ld.b". También me surgió la duda si las imágenes recibidas siempre iban a ser 3x3 en esta subrutina, pero quedo rápidamente contestada al caer en la cuenta de que MFiltro siempre se trata de una matriz 3x3 y de que solo actualiza MFiltro en su llamada.

Por tanto, el 23 rehíce el código de Comp, olvidado el uso de extu y lo dejé todo preparado.

Fecha: 2/10/2018

Asignatura: Arquitectura 88110 Pag: 2

Actualizar Filtro (MFiltro, ModMFiltro)

Si Num o Den = 0, MFiltro se divide igual.
En caso contrario se divide con 2.

$$MFiltro = \begin{pmatrix} a_0 & a_1 & a_2 \\ a_4 & a_5 & a_6 \\ a_7 & a_8 & a_9 \end{pmatrix}$$

ModMFiltro: (Numerador, Denominador)

ACTUALIZA FILTRO: ld r20, r30, 0; pto a málut.

ld r2, r30, 4; Numerador

ld r3, r2, 4; Denominador.

muls r4, r2, r3; Num x Denominador

cmp r7, r4, r0; Si Num o Deno es 0, entonces no se actualiza el filtro.

bbeq eg, r7, cero;

ld r4, r0, 9; r4 es un contador a 9 decimales de la málut.

~~or r4, r20, r20;~~
bucle: cmp r7, r4, r0; } r4 es igual a 0?
bbeq eg, r7, fin; } Si
muls r20, r20, r2; Elimib x Numerador No Se sigue avanzando por la málut
divu r20, r20, r3; (Elimib x No) / Denominador
st r20, r30, 0; Se almacena.

addu r20, r20, 4; Se avanza pointer

~~br r20, bucle;~~

subu r4, r4, 1; Se resta uno "1" al contador.

br bucle; Se salta a bucle.

cero, fin: jmp (r4)



1 palabra = 4 bytes

Diferencia: $\text{Comp}(\text{Imagen1}, \text{Imagen2})$

max $\begin{matrix} 0 & 1 & 2 & 3 \\ 0 & 1 & 2 & 3 \end{matrix}$ 1d.bv.

1. Acumulado inicializado a 0. $r2 \downarrow 8 \text{ bits} \cdot 4 = 32$.

2. Recorre $M \times N$ elementos de cada matriz,

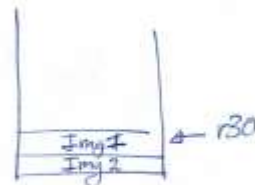
Sumando para cada uno de ellos al acumulador dif el cuadrado del valor de la diferencia entre el pixel img1 y img2

$$\text{Dif} = \text{Dif}_0 + (\text{Imagen1Pixel} - \text{Imagen2Pixel})^2$$

3. Dividir el valor obtenido por $M \times N$.

$$\text{Dif} = \text{Dif} / M \times N$$

4. Devolver por $r29$ el cociente, Dif .



¿Que ocurre si
Imagen1 y Imagen2
tienen distinta dimension?
~~Se~~

Ejemplo: $\begin{pmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{pmatrix} \rightarrow \underbrace{0x04030201}_{1 \text{ byte}}, \underbrace{0x08070605}_{1 \text{ byte}}, \underbrace{0x00000009}_{1 \text{ byte}}$

$4 \text{ bytes} = 1 \text{ palabra}$
 $4 \text{ B} = 4 \cdot 8 = 32 \text{ bits}$

$\begin{pmatrix} 1 & 2 & 3 & 4 \\ 5 & 6 & 7 & 8 \\ 9 & 10 & 11 & 12 \\ 13 & 14 & 15 & 16 \end{pmatrix} \rightarrow \underbrace{0x04030201}_{1B}, \underbrace{0x08070605}_{1B}, \dots, \underbrace{0x0F0E0D}_{1B}$
 $4 \times 4 = 16$

$\begin{pmatrix} 1 \\ 2 \\ 3 \\ 4 \end{pmatrix} \rightarrow 0x04030201$
 $4 \times 1 = 4$

$\begin{pmatrix} 1 & 2 \\ 2 & 3 \\ 3 & 4 \\ 4 & 5 \\ 5 & 6 \\ 6 & 7 \end{pmatrix} \rightarrow 0x03020100, 0x07060504$
 $1B \quad 1B$
 $4 \times 2 = 8$

$\begin{pmatrix} 1 & 2 \\ 3 & 4 \\ 5 & 6 \\ 7 & 8 \\ 9 & 10 \end{pmatrix} \rightarrow 0x04030201, 0x08070605, \dots, 0x00000009$
 5×2

2 bucles anidados: 1º bucle recorre todos los Bytes usados
por la matriz
2º bucle recorre byte

Comp *La simulación recibe imágenes con el*
de. 6 4 4 misma dimensión. ?
 data ~~0x00100000~~ → *data* ~~0x00100000~~ *Despl: data 0x00100000*

Comp: ~~ld~~ *and* r2, r0, r0 ; *Amulador de def.*

~~ld~~ r21, r20, 0 ; *Pto. Jug1*

~~ld~~ r22, r30, 4 ; *Pto Jug2*

~~ld~~ r3, r21, 0 ; *M Jug1*

~~ld~~ r4, r21, 4 ; *N Jug1*

~~ld~~ r5, r21, 0 ; *M Jug2*

~~ld~~ r5, r21, 0 ; *M Jug2*

~~ld~~ r6, r22, 4 ; *N Jug2*

muli r5, r4, r3 ; *hrN contador*

or r6, r5, r5 ; *Copie de r5.*

bucle : *comp* r7, r6, 0 ;

bbl eq, r7, r6, *bucle ;*

subu r6, r6, 1 ;

~~extu~~ ~~r2, r21,~~

~~extu~~ ~~r4, r22,~~

bucle - int : *ld* r8, r8, *Despla ;* *Cargamos en*
r8 displ.

bucle - int *addu* r9, r9, 1 ;

extu r10, r10, r8 ; *obtenemos el valor del pixel.*

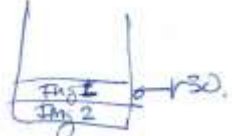
extu r11, r11, r8 ;

addu r2, r10, r11 ; *PixelJug1 + PixelJug2.*

subu r6, r6, 1 ; *-1 contador de elementos.*

subu r8, r8, 2 ; *se desplaza 2*

~~comp~~



r2 = def.
r21 = Jug1
r22 = Jug2.
r3 = M Jug1
r4 = N Jug1
r5 = ~~M Jug2~~ A x N
r6 = contador bucle.
r7 = cont. int.

bits = 9-5 longitud. → *Sec 2*

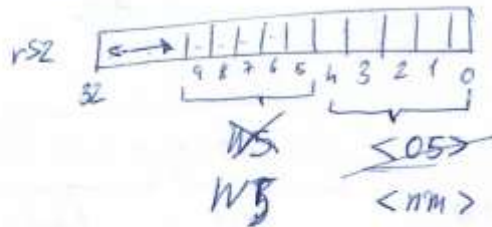
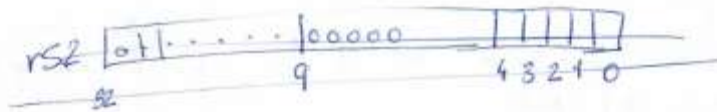
bits = 4-0 Desplazamiento.

Hay que avanzarlo

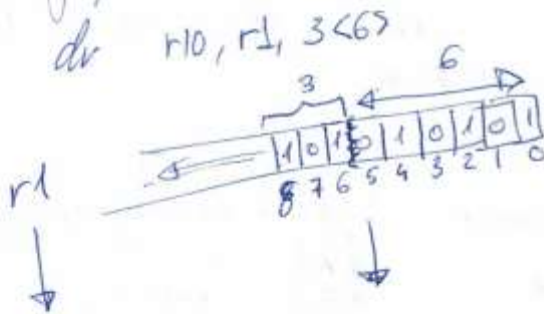
comp r7, r9, 4 ;
bbl ne, r7, *bucle - int*
ld r4, r0, r0 ;
br *bucle ;*

Hay que usar esto

~~dr.~~
dr rd, rsl, rs2.



Ejemplo: dr → Lee rsl y usa un campo de "0"
en el rd, Longitud = NS
Desplaz = OS.



Si OS = 0, usa los 32 bits.

COMP

Despla: ^{W5} ^{C5}
data 0x00011111

COMP: ~~ld~~ and r2, r0, r0; Anulador dif

ld r21, r30, 0; Pto. Img1

ld r22, r30, 4; Pto. Img2

ld r3, r21, 0; H Img1

ld r4, r21, 4; N Img1

ld r12, r0, r0;

ld r9, r0, r0; Contador interno

movl r5, r4, r3; MrN contador

or r6, r5, r5; Copia contador MrN.

buck: cmp r7, r6, r0;

bblt ef, r7, fin-buck;

ld r8, r8, Despla

buck-int: addl r5, r9, 1;

extr r10, r20, r8;

extr r11, r21, r8;

subl r12, r10, r11;

subl r6, r6, 1;

subl r8, r8, 2;

cmpl r7, r9, 4;

bblt ne, r7, buck-int;

ld r9, r0, r0;

br buck;

Como cargo iba de el
segunda? se supone que fue
como un dir. ~~pero luego~~
¿?

Buck para recorrer los 4 canales
de la ~~para~~ ~~matriz~~ que caben en
4 byte.

$$* Dif = Dif_0 + (Img1 - Img2)^2$$

obtenemos los datos
= suma de los
2 imagenes.
dif =

* subu ~~addu~~
subu r12, r12, r12
addu, r2, r2, r12

27/10/2018 y 28/10/2018 – 2-3 horas cada día

Se realizaron cambios en las subrutinas luego de pasar una corrección el 24 de octubre.

Se prepararon algunas pruebas y test individuales para cada subrutina.

29/10/2018 y 30/10/2018 – 2-3 horas cada día

El resultado de la corrección dio como resultado que las 3 subrutinas para el 1º hito pasaban las pruebas. En estos días se empezó a desarrollar las subrutinas para el 2º hito. Estas subrutinas requirieron de mas tiempo para corregirse y completarse para que pasarán el 2º hito.

Para este día se envió el código para que se corrigiera junto al 1º hito con el resto de subrutinas.

Primero empecé con “ValorPixel”, que resulto ser sencilla de realizar. Esta se realizó directamente en PC y no requirió demasiadas líneas de código para completarla.

Por otra parte, “Submatriz” demostró ser algo mas complicada a la hora de plantear como ir moviéndose por cada matriz y como extraer submatrices de matrices con dimensiones superiores a 5x5.

El corrector dio como resultado el 31:

ValorPixel = 9 de 10 fallos.

Submatriz = 10 de 10 fallos.

Lo que me llevo a plantearme rehacer el código de ambas rutinas.

31/10/2018 – 1 hora

El corrector dio como resultado el paso del 1º hito y los errores ya comentado anteriormente. Los errores de “ActualizaFiltros” se solucionaron como se comentó anteriormente cambiando mulu y divu por muls y dvs respectivamente.

4/11/2018 – 3-4 horas

Se repasó y se rehizo el código de “ValorPixel” y “SubMatriz”.

“ValorPixel” estaba bien pero al igual que ocurrió en “ActualizaFiltro” cometí el error de olvidar que se trataban de enteros con signo en las operaciones de recogida de datos (ld.b) y con el muls.

Estos errores se pueden decir que son heredados de las 3 primeras subrutinas realizadas.

La subrutina se volvió a hacer desde 0 para coger nuevas ideas y realizarse de una manera completamente distinta. El problema era como lidiar con la matriz interna cuando se elegía un elemento del interior. La solución elegida ha sido crear 2 punteros para el caso interno.

Para r20 siempre apuntaría al inicio de Imagen. Mientras que r21 será el puntero de la submatriz de Imagen. Para ello usamos el puntero r20 como puntero de referencia para calcular el inicio de cada fila de la submatriz y esto unido a 2 bucles, uno para recorrer los 3 elementos internos y el otro para recorrer las 3 filas y calcular el desplazamiento necesario.

El caso de un elemento en el borde se trata de un caso muy sencilla, ya que solo necesitamos obtener el elemento central.

5/11/2018 – 2-3 horas

Realizados casos de prueba y testeados con ejemplos propios y los suministrados por la asignatura.

Todos parecen funcionar de acuerdo a lo esperado. Se ha realizado la entrega para el corrector.

6/11/2018 – 3 horas

El corrector da como resultado que “ValorPixel” pasa las pruebas, pero “SubMatriz” falla en 3.

Los 3 fallos se dan en matriz de tipo 5x5 o superiores. He creado pruebas para simular este tipo de matrices al estilo de los casos fallidos.

El error se debía a la forma de calcular el desplazamiento entre filas.

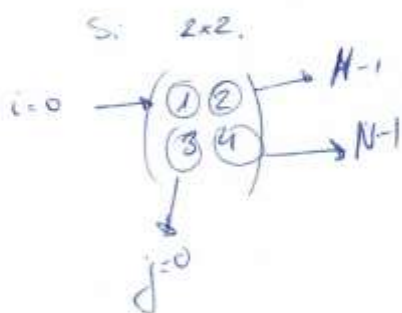
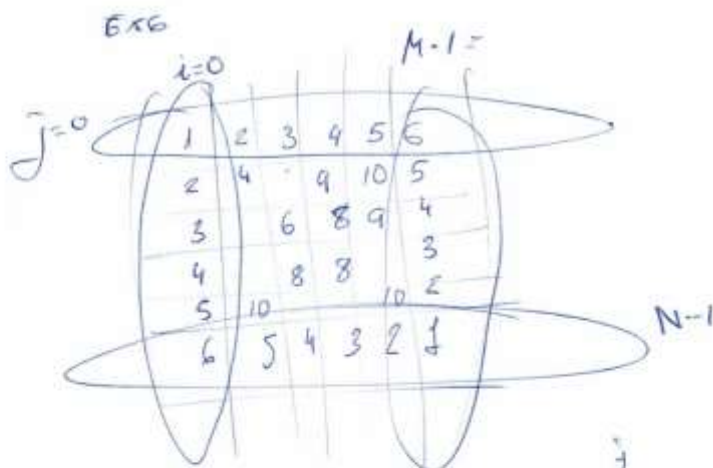
Lo hacia usando $M \times i$, pero debía ser $N \times i$, debido a que debo calcular cuantos elementos de la fila multiplicándolo por las filas hasta el elemento al que queremos llegar. Por tanto, en cada iteración del bucle exterior calculamos el desplazamiento hasta el 1º elemento de la fila de la submatriz.

Con esto ya tendría todo preparado para superar el 2º hito. Los días siguientes empecé a realizar las siguientes rutinas en el orden citado a continuación:

1. “FilPixel”
2. Filtro
3. FiltRec

Fecha: 24/10/2018

Asignatura: Programa 88110 Pag: 6



$i \downarrow j \rightarrow$

$\begin{pmatrix} 1 & 2 & 5 \\ 4 & 5 & 6 \\ 2 & 8 & 9 \end{pmatrix} \rightarrow \begin{matrix} 0x04030201, 0x08070605 \\ 0x00000009 \end{matrix}$

$i=1$

$j=2$

~ 6

Imagen = $H, N, \text{Máxiz}$ / Parametro Entrada

SubImg = $H=3$ / Parametro salida.

$N=3$

20 de agosto

$\begin{matrix} 4 & 2 & 2 & 4 \\ 0x04030201, & 0x08070605, & 0x00000009 \end{matrix}$

$\begin{matrix} \text{Fila 1} & \text{Fila 0} & \text{Fila 2} & \text{Fila 1} & \text{Fila 2} \end{matrix}$

Meluz $\rightarrow M=3, N=3$

~~2.~~ ~~ptrM~~ ptrM, ptrN, i, j.

i = 1. ptrM = 0
j = 2. ptrN = 0.

r20 = puntero a Meluz.

Bucle Filas: $\text{cmp ptrM} = i$

bbl eg, fila encontrada.

~~addu ptrM + 1~~ Avanza el puntero

~~addu~~

~~addu r20, r20, 3;~~ Avanza 4 fila el puntero

bn BucleFilas

0x04030201

ptr
inicial

0x04030201

ptr

Fila ENCONTRADA. $\text{cmp ptrN} = j$

bbl eg, fila encontrada.

~~addu ptrN + 1~~

~~addu r20, r20, 2;~~ Avanza

mulu r5,

Esto se puede obtener con el i y j que nos dan.

$\text{cmp ptr} = i$

bbl eg, fila encontrada.

mulu r6, i, 3;

M	i	Desplazamiento
3	0	0
3	1	3
3	2	6
3	3	9

N	j
3	0
3	1
3	2

que este caso basta

con separar el desplazamiento de j.

Calcular desplazamiento en filas (M.i)

Calcular desplazamiento en columnas (N.j)

$$\begin{array}{l} r4 = 5 \\ r2 = 1 \end{array} \quad \begin{array}{l} r5 = 5 \\ r3 = 0 \end{array}$$
$$r_2 = 1 \quad r_3 = 0.$$
$$1. \quad 5.4 = 5.$$
$$1. \quad 5 + 5 = 5.$$
$$= \frac{10000}{100 + 5} = 100 \text{ position}$$
$$= \text{ddo } 70, 20, 10$$

100-103

add 120, 120, 13

105

Sigumite Arcto. fol.

10/10/2019

friday

add $r_2, r_3, t.$

 $i+1=2.$

mule r6, r4, —→

$$5 - 0 = 5 \text{ elements west side.}$$

5/15/70

addn 170, 170, 176.

Wm. W. B. B. B.

 $57 = 5 \cdot 5$ 

072

136-75 i y j, vertical fixed sample.

$i \rightarrow j$ } \rightarrow Apunta al primer pixel de la

J-65
pneu pied de la machine 3x3.

$\begin{pmatrix} 7 & 8 & 9 \\ 12 & 13 & 14 \\ 17 & 18 & 19 \end{pmatrix}$

28/3/28

$$12 \rightarrow i=2, j=4.$$

* llygo r9=0.

sub vs, rs, d.
also vs vs + not vs¹²
also
vs vs,

~~Subv 66, 69, 72, 75, 78~~
$$r_{\text{mole}} = r_6, r_4, r_2; M \times i$$

add r29, r20, r0

$$m, n, r, s, N \times 2$$

134216, n

addw r28, r28, r3; # 148

$$\begin{cases} x-1=4 \\ y-1=0 \end{cases}$$
$$f = 1 - 1 = 0.$$

Duplex.
in files.

Dispute
in column.

Fecha: 29/10/2018

Asignatura: ES/10

Pag: 8

if: $\begin{cases} i = \text{fil.} \\ j = \text{columna.} \end{cases}$

Asignatura: 88110 Pag: 9

$$\begin{pmatrix} 11 & 12 & 13 \\ 21 & 22 & 23 \\ 31 & 32 & 33 \end{pmatrix} \begin{matrix} \rightarrow 3 \text{ elementos, } \cancel{1} \text{ fila.} \\ \rightarrow \text{Si se elige la } 22. \\ \downarrow \\ 2 \text{ filas. } 2 \text{ cols.} \\ 2 \cdot 3 = 6 \text{ elementos.} \end{matrix}$$

Si se elige la 22..

-4 elementos dentro y +4 de

Ejemplo: $A = \begin{pmatrix} 2 & 4 & 3 \\ 1 & 2 & 4 \\ 5 & 7 & 3 \end{pmatrix}$

Si elegimos A.

$$\begin{pmatrix} 2 & 4 & 3 \\ 1 & 2 & 4 \\ 5 & 7 & 3 \end{pmatrix}$$

Se copia la matriz 3x3 alrededor del actual.

Si elegimos B.

$$\begin{pmatrix} 2 & 4 & 3 \\ 5 & 5 & 4 \\ 5 & 5 & 3 \end{pmatrix}$$

Los números alrededor se sustituyen con el actual.

Fecha: 29/10/2018

Asignatura: 89110

Pag: 10

$M = \text{Filas}, N = \text{Columnas}$

5x8

r20	0	1	2	3	4	5	6	7
0	0	1	2	3	4	5	6	7
1	8	9	10	11	12	13	14	15
2	16	17	18	19	20	21	22	23
3	24	25	26	27	28	29	30	31
4	32	33	34	35	36	37	38	39

Arrows pointing to cells (3,4)=28, (4,3)=35, (4,7)=39 with labels "FALLA".

Si desinose el 30.

$i=3, j=5$

$M \cdot i \rightarrow 8 \cdot 3 = 24$

j : fila submatriz.

$i-1=2, j-1=4$

3º
Para 40 (4,7)

Submatriz

$\begin{pmatrix} 40 & 40 & 40 \\ 40 & 40 & 40 \\ 40 & 40 & 40 \end{pmatrix}$

1º
Para 35 (4,2)

Submatriz

$\begin{pmatrix} 35 & 35 & 35 \\ 35 & 35 & 35 \\ 35 & 35 & 35 \end{pmatrix}$

2º
Para 30 (3,5)

$\begin{pmatrix} 24 & 22 & 23 \\ 29 & 30 & 31 \\ 37 & 38 & 39 \end{pmatrix}$

Kodijo \rightarrow 0x1284

Dir: 0x2A24, Activa

Dir. SubInt: 0x2A4C, SubInten

$i: M=5, N=8$

5.

P+ 1352

Back point
para st.b n0, r21, 0
P+ 1392

(4,7).

Matriz 5x8.

1. $4 \cdot 8 = 32$.

2. $+7$

3. 39 .

Matriz 3x5

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

$12 \rightarrow i=2, j=4$.

$3 \cdot 2 = 6$

$i=1, j=1$

3x3

	0	1	2
0	1	2	3
1	4	5	6
2	7	8	9

$8, i=2, j=2$
 $3 \cdot 2 = 6$

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

Matriz 5x3

$12 \rightarrow i=3, j=2$.

BZBUS

31

$3 \cdot 4 = 3$ elementos

14/11/2018 – 3 horas

Primeras pruebas a "FilPixel" y correcciones de "SubMatriz" comprobando que funcionan bien cuando son llamadas desde otra rutina.

FilPixel es el 1º ejemplo en el que compruebo que al llamar a SubMatriz no funciona correctamente debido a que pierdo en algún punto del programa el valor de retorno. Aquí es cuando empiezo a incluir la dirección de retorno y a gestionar la pila. Las pruebas usadas para testear "FilPixel" además prueban si funciona correctamente "SubMatriz" y "ValorPixel" cuando se hace la llamada a "FilPixel".

FilPixel prueba 2:

Matriz 5x5.

MFilter 3x3

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

$$\text{-- IMAGEN --}$$

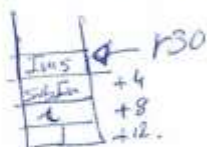
$$\begin{matrix} & 0 & 1 & 2 & 3 & 4 \\ \begin{matrix} 0 \\ 1 \\ 2 \\ 3 \\ 4 \end{matrix} & \begin{pmatrix} 11 & 12 & 13 & 14 & 15 \\ 21 & 22 & 23 & 24 & 25 \\ 31 & 32 & 33 & 34 & 35 \\ 41 & 42 & 43 & 44 & 45 \end{pmatrix} \end{matrix}$$

Como dice
funcionan y
que salida
se espera.

Pixel seleccionado: $[2,2] = 23$

1° Submatriz.

(Imagen, Subimg, i, j)



Submatriz → Valor.
3x3.

$$\begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

Submatriz iter
a memoria en FilPixel.

se elige el $[2,2] = 23$.

Submatriz → $\begin{pmatrix} 2 & 3 & 4 \\ 22 & 23 & 24 \\ 32 & 33 & 34 \end{pmatrix}$ Resultado que se debe obtener.

2° Valor Pixel

(Submatriz, MFilter)

$$Acc = 2 \cdot 0 + 3 \cdot 0 + 4 \cdot 0 + 22 \cdot 0 + 23 \cdot 1 + 24 \cdot 0 + 32 \cdot 0 + 33 \cdot 0 + 34 \cdot 0$$

→ $Acc = 23$ Resultado final a Valor Pixel.

Este Pixel no será guardado y será el pixel final.

3^o FilPredCon $r29 = 23 \rightarrow VP$ antitiltio = 1. \rightarrow Peso

$$\frac{23}{1} = 23 \text{ valor del pixel. No se ajusta}$$

Se devuelve por r29 el valor del pixel

fultado $r29 = 23$

▣ Dirección en memoria de las instrucciones.

$$\text{FilPred: } \text{BdH PC} = 1484 \quad \left| \quad \frac{\Delta \text{HORA}}{\text{PC} = 1500}$$

salto a subitua: PC = 1556, subitua: PC = 1284

fin subitua: PC = 1560

al lago de subitua, PC = 1592

Puntos a Subirgen
 \rightarrow 0xEA3C

FilP.
 14 = 53C4 \rightarrow r4 = 618

Subitua

• ~~FinalPixel: PC = 1500.~~

4 Bytes = SubIMAGEN = 0x1E43C

• ~~SALTO A SubPixel: PC = 1588.~~

~~SubPixel: PC = 1584 1300~~

• ~~Fuori da SubPixel: PC = 1582.~~

~~PC = 1456, br FIN.~~

~~PC = 1582, Jump r4.~~

• ~~SALTO A ValorPixel: PC = 1624~~

~~ValorPixel: PC = 1230~~

~~FinalPixel: PC = 1500~~

~~br SubPixel: PC = 1592~~

~~PC+4 = 1596~~

~~br ValorPixel: PC = 1644~~

59968 - " | " | 10000000 | 04530000 |
59984 - 28200000 | 01000000 | 01000000 | 4C2B0000 |
IMAGEN | i | j | MFILTRO

Fecha: 14/11/2018

Asignatura: 1 88110 Pag: 14

FILPIXEL: PC = 1516

SubJug: 0xEABC

SALTO a Submá/n7: PC = 1592

SALTO a RdxRxd: PC = 1644

FIN FILPIXEL: PC = 1756. ^{ANTES} → of R9, r2, r2, PC = 1736

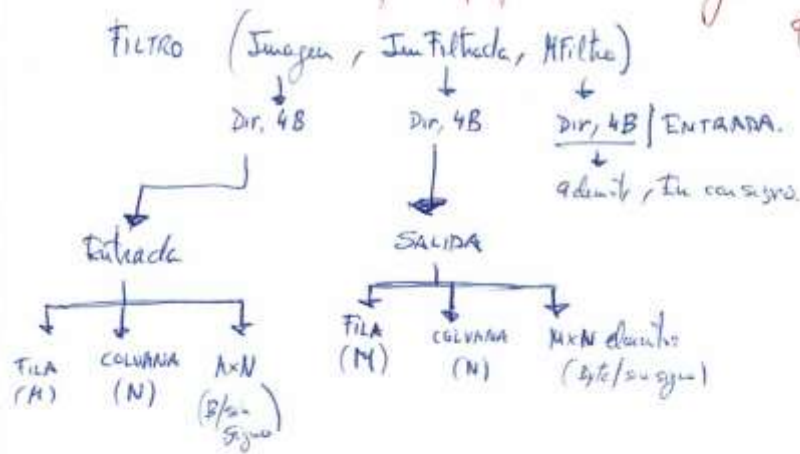
15/11/2018 – 3 horas

En este día se prepara el código de “Filtro” y “FiltRec”, se implementa en papel el código de “Filtro” y se deja listo el de “FiltRec” para su desarrollo.

La implementación de “Filtro” no resulto difícil de sacar, teniendo en cuenta que hay que usar el algoritmo propuesto. Lo primero que hago es preparar todos los parámetros que serán necesarios para las llamadas a “FilPixel”.

Se realiza mediante 2 bucles, que van avanzar i y j, que son 2 parámetros que debe recibir “FilPixel” para seleccionar el pixel que va a filtrarse. Al final obtenemos una imagen filtrada.

2 horas
para preparar el código de Filtro y FiltRec.



CURSOS	← 130, 132
Dir. Rec	+4
IMAGEN	+8
ImFiltrada	+12
HFILTER	+16

1. Copiar M, N de Imagen en ImFiltrada.

2. Desde el punto $i=0$ hasta $i=(M-1)$

Recorriendo la matriz en columnas, $j=0$ hasta $j=(N-1)$

2.1. Preparación de parámetros, y llamo a

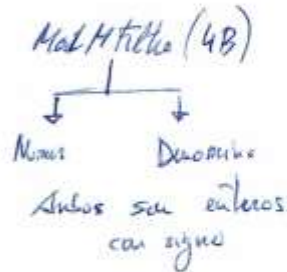
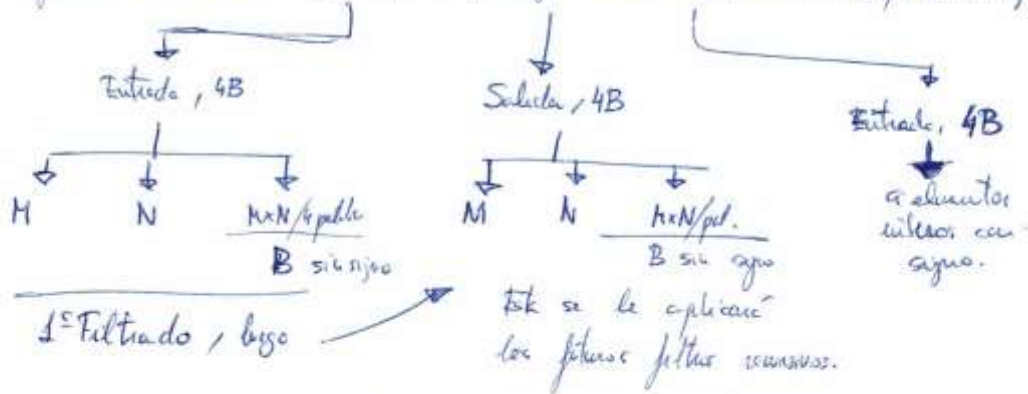
FiltRec (Imagen, i, j, HFilter)

2.2. Aplicar en (i, j) de ImFiltrada, el valor devuelto por FiltRec en r29.

3. Retornar al llamante al finalizar ambos bucles.

Filtro recursivo

Difusionia = FiltRec (Imagen In, Funcion Out, HFilter, ModHFilter, Ncanbios)



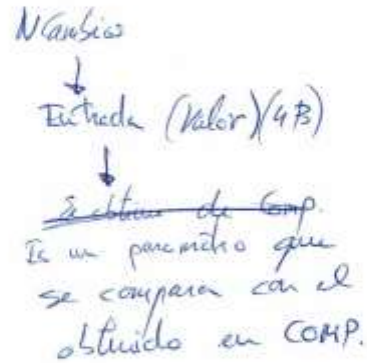
Difusionia

↓

Salida por r29

↓

Entero con signo



¿Cómo calcular la imagen? el tamaño de la imagen?

$$\begin{array}{l}
 3 \times 3 \\
 \hline
 0x01223344, 0x55667788, 0x99 \\
 \hline
 3 \text{ bytes}
 \end{array}
 \quad
 \begin{array}{l}
 3 \times 5 = 15 \quad 16 \\
 \hline
 3 \quad 16 \\
 5 \quad 4 \\
 \hline
 12
 \end{array}$$

$$\begin{array}{l}
 2 \times 4 \\
 \hline
 0x11223344 \quad 0x55667788 \\
 \hline
 2 \text{ bytes}
 \end{array}
 \quad
 \begin{array}{l}
 8 \quad 16 \\
 \hline
 0 \quad 8 \\
 \hline
 8
 \end{array}$$

$$\begin{array}{l}
 5 \times 8 = 40 \text{ bits} \\
 \hline
 40 \quad 16 \\
 00 \quad 10B
 \end{array}$$

$$\begin{array}{l}
 3 \times 5 = 15 \quad 16 \\
 \hline
 3 \quad 16 \\
 5 \quad 4 \\
 \hline
 6
 \end{array}$$

HAL



Filtro: Push (rt)

Push (r31)

or r31, r30, r30.

ld r20, r31, 5 ; Puntos a Juegan

ld r2, r20, 0 ; M sub r4, r2, 1 ; M-1

ld r3, r20, 4 ; N sub r5, r3, 1 ; N-1

addu r20, r20, 8 ; Puntos al f° dentro de Juegan.

~~ld r4, r0, r0 ; i, se preparan ambas contadores~~

~~or r5, r0, r0 ; j, para recorrer todos los elementos de la matriz.~~

~~No necesitamos r0.~~

~~sub r4, r2, r3 ; M x N = Total elementos → Contador.~~

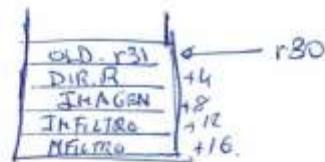
~~Bucle f: ~~ld cap, r7, r4, r0 ; Total elem. Recorre == 0?~~~~

~~ld hbt, eq, r7, r0 - Bucle~~

~~or r4, r0, r0 ; i, los dos contadores que recorren para Filtrado.~~

~~or r5, r0, r0 ; j~~

filter: Push(r2)
Push(r31)



or r31, r20, r30.

ld r20, r31, 8 ; IMAGEN

ld r2, r20, 0 ; H

ld r3, r20, 4 ; N

ld r21, r31, 12 ; INFILTR0

ld r27, r31, 16 ; MFILTR0

~~sub r4, r2, 1 ; M-1~~

~~sub r5, r3, 1 ; N-1~~

← No se usa

or r8, r0, r0 ; i → ~~Debe compararse con 1~~

or r9, r0, r0 ; j

Bucle-H: ~~add r7, r8, r2 ; i++~~ ~~cmp r7, r8, r2 ; i == M?~~

bbl r9, r7, FIN-Bucle-H

~~add r8, r8, 1~~

or r9, r0, r0 ; j = 0

Bucle-N: ~~br Bucle-N~~ ~~cmp r7, r9, r3 ; j == N?~~

bbl r9, r7, Bucle-H

Push(r22) ; MFILTR0

Push(r9) ; j

Push(r8) ; i

Push(r20) ; IMAGEN

bsr FilPixel;

Pop(r20)

Pop(r8)

Pop(r9)

Pop(r22)

Cambiando
esto de
posición,
luego del bucle-N

Falta guardar
variables usadas
como r2, r3 (M y N)
además del puntero de
INFILTR0

st r29, r21, 0 ; Carga el valor de r29 = VPixel
en Intfilho.

addu r9, r5, t ; j++;

addu r21, r21, t ; ++ el puntero Intfilho.

~~addu r21, r21, t ; ++ el puntero Intfilho.~~

exp r21, r21

br bnde - N.

→ El puntero
que paso
siempre cita
al principio.

FIN-Bnde - H : Pop(r31) ; Recuperación de r31.
Pop(r1) ; Recuperación de dir de retorno
Jmp(r4) ; ~~Salto~~ Salto a dir de retorno

Notas: Hay que asegurarse que el bucle exterior
funciona.

Solución: ~~Antes~~ ~~se ordena~~ Pasar el código
del bucle exterior debajo del código del
bucle interno.

Ahora

Bucle - E : { jmp FIN

Bucle - I : { cap Bnde - E
br Bnde - I

Luego

Bucle - I : { cap Bnde
br Bnde - I

Bucle - E : { cap FIN
br Bnde - I

16/11/2018 – 2 - 3 horas

Se implementa en papel la rutina “FiltRec”. Posteriormente se pasan a código “Filtro” y “FiltRec”.

17/11/2018 – 4 horas

Se realizan una batería de pruebas para “Filtro” y “FiltRec”. Estas pruebas dan resultado adecuados.

Aunque el funcionamiento final de “FiltRec” todavía no parecer funcionar adecuadamente.

Todas las pruebas se hacen a mano para prever cual debería ser el resultado dado en cada prueba.

Fecha: 16/11/2018

~~NO~~ YA NO

Asignatura: 88110

Pag: 20

Salida de la neuronalidad: 1. $i \neq$ maximo de veces que se puede filtrar
2. Si $Comp < NCarbhos$.

FiltRec: Push (rd)
Push (r31)
or r31, r30, r30.
subu r30, r30, (r28) → r28 = ImageTAP
mulu r2, r2, r3; MxN → r28 va a tener el calculo neuronal neuronal XB
or r5, r4, r4
Push {
bstr Filtro; (ImageFu, ImageOut, AFiltro)
Pop {
Push {
bstr Actualizafiltro; (AFiltro, ModAFiltro)
Pop {
cap r5, r0; salto si r5=0 a caril.
Bucle-Almacenas: stb r22, r28, 0; r22 = Pt ImageOut
r28 = Pt ImageTAP
addu r22, r22, 1
addu r28, r28, 1
subu r5, r5, 1; (MxN)--
br Bucle-Almacenas
Cont-5NF: bstr ntifiltros; (opac = -1) Decremento mayor
bbl r7, r29, r0
cap r7, r7, fin-kearst
Fin-Recal: or r29, r29, -1 → r29 spots
brr FIN

old.rsi	← r30
clon.net	+4
ImageFu	+8
ImageOut	+16
AFiltro	+16
ModAFiltro	+20
NCarbhos	+24

Push {

Comp (Image IN, ImageOut); r29
Pop {

cmp r7, r29, r6 ; r29 < r6; comp < NConsus
bbl lt, r7, Fin ~~Addr~~

o El caso continuo transición.

Push {

bsr FiltRec (Img TMP, Img Out, HFilter, HndHFilter, NConsus)

Pop {

Fin-Rec + : or r29, r29, #1

Fin : Pop(r31)
Pop(r7)
Jmp(r7)

18/11/2018 - 2 horas

Empiezo a preparar la memoria y a rellenarlas con los días ya pasados.

28/11/2018 – 2 horas

Asistencia a tutoría y testeo de “Filtro”

Encontré un error en el bucle exterior que llama “FilPixel”. Este error consistía en que el bucle realizaba una interacción más pasándole a “FilPixel” un valor “i” no valido y que esta fuera de la matriz.

La solución del error consiste en mover “addu” que incrementaba la variable “i” antes de la comparación para saber si se habían recorrido todas las filas de la matriz.

29/11/2018 – 2 horas

“Filtro” ya parece pasar todas las pruebas suministradas en la asignatura, mostrando los datos como en las pruebas.

“FiltRec” tenía varios fallos. No demasiado graves y con fácil solución:

1. Faltaba un bucle que agregaré correctamente el valor de M y N a “ImagenTMP”. Este error provocaba que la “ImagenTMP” no tuviera las dimensiones de la matriz y además, su dimensión no correspondiera con la esperada.
2. Se cargaba mal “ImagenTMP” usando un Id. La solución pasaba por hacer una copia con “or” del puntero r30 en el registro que uso como “ImagenIn” al pasársela a “FiltRec” en su llamada recursiva.
3. Por último, algunos fallos al realizar los “testeos”, no fijándome bien en que “NCambios” o “nF” debían tener cada prueba.

30/11/2018 – 2 horas

Ultimas comprobaciones para subir el código al corrector de la asignatura. Todavía me quedan 2 correcciones. La memoria también quedará finalizada hoy si el corrector da un resultado bueno.

1/12/2018

Finalmente, el resultado del correcto ha sido positivo. Mi código supera todas las pruebas del correcto de la asignatura y las pruebas de los casos de ejemplos suministrados por la asignatura.

2. Pruebas

```
; nF = nFiltrados(oper)
```

```
oper1:    data 5
oper2:    data 0
oper3:    data -5
oper4:    data -10
oper5:    data 14
oper6:    data -1
```

```
-----
          or r2,r2,10      ; Inicializo la variable nF = 10
          st r2,r0,0

          or r2,r0,r0
          LOAD(r2,oper1)   ; oper = 5
          PUSH(r2)

          or r2,r0,r0      ; Esto vacia r2 para visualizarlo mejor cuando entra en la
                          ; subrutina y ver que se carga bien el dato desde la pila.
          bsr nFiltrados

Test2NFI:
          or r2,r2,0       ; Inicializo la variable nF = 0
          st r2,r0,0

          or r2,r0,r0
          LOAD(r2,oper2)   ; oper = 0
          PUSH(r2)

          or r2,r0,r0      ; Esto vacia r2 para visualizarlo mejor cuando entra en la
                          ; subrutina y ver que se carga bien el dato desde la pila.
          bsr nFiltrados

Test3NFI:
          or r2,r2,7       ; Inicializo la variable nF = 7
          st r2,r0,0

          or r2,r0,r0
          LOAD(r2,oper3)   ; oper = -5
          PUSH(r2)

          or r2,r0,r0      ; Esto vacia r2 para visualizarlo mejor cuando entra en la
                          ; subrutina y ver que se carga bien el dato desde la pila.
          bsr nFiltrados

Test4NFI:
          or r2,r2,0       ; Inicializo la variable nF = 0
          st r2,r0,0

          or r2,r0,r0
          LOAD(r2,oper4)   ; oper = -10
          PUSH(r2)

          or r2,r0,r0      ; Esto vacia r2 para visualizarlo mejor cuando entra en la
                          ; subrutina y ver que se carga bien el dato desde la pila.
          bsr nFiltrados

Test5NFI:
          or r2,r2,0       ; Inicializo la variable nF = 0
          st r2,r0,0

          or r2,r0,r0
          LOAD(r2,oper5)   ; oper = 14
          PUSH(r2)

          or r2,r0,r0      ; Esto vacia r2 para visualizarlo mejor cuando entra en la
                          ; subrutina y ver que se carga bien el dato desde la pila.
          bsr nFiltrados

Test6NFI:
          or r2,r2,14      ; Inicializo la variable nF = 14
          st r2,r0,0

          or r2,r0,r0
          LOAD(r2,oper6)   ; oper = -1
          PUSH(r2)

          or r2,r0,r0      ; Esto vacia r2 para visualizarlo mejor cuando entra en la
                          ; subrutina y ver que se carga bien el dato desde la pila.
          bsr nFiltrados
```

```

;*****
; Numerador y denominador para ActualizarFiltro
;*****
; ActualizaFiltro(MFiltro, ModMFiltro)

MATRIZFILTRO1:
    data 1,2,3
           ,4,5,6
           ,7,8,9
NumDem1:      data 4,2

MATRIZFILTRO2:
    data 2,4,1
           ,3,5,7
           ,6,8,9
NumDem2:      data 2,4

MATRIZFILTRO3:
    data 4,5,6
           ,7,8,9
           ,1,2,3
NumDem3:      data 1,2

MATRIZFILTRO4:
    data 4,5,6
           ,7,8,9
           ,1,2,3
NumDem4:      data 0,2

```

```

;*****;
; Pruebas para ActualizarFiltro
;*****;
; ActualizaFiltro(MFiltro, ModMFiltro)
Test1ActFil:
    LEA(r2,NumDem1)
    PUSH(r2)
    LEA(r2,MATRIZFILTRO1)
    PUSH(r2)

    or r2,r0,r0      ; Esto vacia r2 para visualizarlo mejor cuando entra en la
                    ; subrutina y ver que se carga bien el dato desde la pila.
    bsr ActualizaFiltro

Test2ActFil:
    LEA(r2,NumDem2)
    PUSH(r2)
    LEA(r2,MATRIZFILTRO2)
    PUSH(r2)

    or r2,r0,r0      ; Esto vacia r2 para visualizarlo mejor cuando entra en la
                    ; subrutina y ver que se carga bien el dato desde la pila.
    bsr ActualizaFiltro

Test3ActFil:
    LEA(r2,NumDem3)
    PUSH(r2)
    LEA(r2,MATRIZFILTRO3)
    PUSH(r2)

    or r2,r0,r0      ; Esto vacia r2 para visualizarlo mejor cuando entra en la
                    ; subrutina y ver que se carga bien el dato desde la pila.
    bsr ActualizaFiltro

Test4ActFil:
    LEA(r2,NumDem4)
    PUSH(r2)
    LEA(r2,MATRIZFILTRO4)
    PUSH(r2)

    or r2,r0,r0      ; Esto vacia r2 para visualizarlo mejor cuando entra en la
                    ; subrutina y ver que se carga bien el dato desde la pila.
    bsr ActualizaFiltro

```

```

; *****
; Datos para COMP
; *****
; diferencia = Comp ( Imagen1, Imagen2)

; IMG1 --> | 1 2 3 |
;          | 4 5 6 |
;          | 7 8 9 |

; IMG2 --> | 9 8 7 |
;          | 6 5 4 |
;          | 3 2 1 |

IMG1:      data 3,3,0x04030201,0x08070605,0x00000009
IMG2:      data 3,3,0x06070809,0x02030405,0x00000001

; -----;

; IMG3 --> | 4 3 |
;          | 2 1 |

; IMG4 --> | 1 2 |
;          | 3 4 |
;          | 1 2 |
;          | 3 4 |

IMG3:      data 2,2,0x01020304
IMG4:      data 4,2,0x04030201,0x04030201

```

```

; *****;
; Pruebas para Comp
; *****;
; diferencia = Comp ( Imagen1, Imagen2)
Test1Comp:
    LEA(r2,IMG2)
    PUSH(r2)
    LEA(r2,IMG1)
    PUSH(r2)

    or r2,r0,r0      ; Esto vacia r2 para visualizarlo mejor cuando entra en la
                     ; subrutina y ver que se carga bien el dato desde la pila.
    bsr Comp;

Test2Comp:
    LEA(r2,IMG4)
    PUSH(r2)
    LEA(r2,IMG3)
    PUSH(r2)

    or r2,r0,r0      ; Esto vacia r2 para visualizarlo mejor cuando entra en la
                     ; subrutina y ver que se carga bien el dato desde la pila.
    bsr Comp;

```

```

;*****
;Datos para ValorPixel
;*****
; VPixel = ValorPixel (SubImg , MFiltro)
;
;
; MFiltro1 -->| 1 1 1 |
;              | 1 0 1 |
;              | 1 1 1 |
;
; SubImg1 -->| 1 2 3 |
;            | 4 5 6 |
;            | 7 8 9 |
;
;
MFiltro1:  data 1,1,1,
           1,0,1,
           1,1,1

SubImg1:   data 0x04030201,0x08070605,0x00000009

           ;-----;

; MFiltro2 -->| 0 0 0 |
;              | 0 1 0 |
;              | 0 0 0 |
;
; SubImg2 -->| 0 0 0 |
;            | 0 55 0 |
;            | 0 0 0 |
;
;
MFiltro2:  data 0,0,0,
           0,1,0,
           0,0,0

SubImg2:   data 0x00000000,0x00000055,0x00

; Resultado en r29 = 0x55, (85 decimal)
           ;-----;

; MFiltro3 -->| 0 0 0 |
;              | 0 -254 0 |
;              | 0 0 0 |
;
; SubImg3 -->| 0 0 0 |
;            | 0 55 0 |
;            | 0 0 0 |
;
;
MFiltro3:  data 0,0,0,
           0,0xFFFFFFE,0,
           0,0,0

SubImg3:   data 0x00000000,0x00000055,0x00

; Resultado en r29 = 0xFF FF FF 56, (-170 decimal)
           ;-----;

```

```

; MFiltro4 -->| -254 -254 -254 |
;              | -254 0 -254 |
;              | -254 -254 -254 |
;
; SubImg4 -->| 10 11 12 |
;            | 13 14 15 |
;            | 16 17 18 |
;
;
MFiltro4:  data 0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF,
           0xFFFFFFFF,0,0xFFFFFFFF,
           0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF

SubImg4:   data 0x13121110,0x17161514,0x18

; Resultado en r29 = 0xFF FF FE C0, (-320 decimal)

```

```

;*****
; Pruebas para ValorPixel
;*****
; VPixel = ValorPixel (SubImg , MFiltro)
Test1Pixel:
    LEA(r2,MFiltro1)
    PUSH(r2)
    LEA(r2,SubImg1)
    PUSH(r2)

    or r2,r0,r0      ; Esto vacia r2 para visualizarlo mejor cuando entra en la
                    ; subrutina y ver que se carga bien el dato desde la pila.
    bsr ValorPixel

Test2VPixel:
    LEA(r2,MFiltro2)
    PUSH(r2)
    LEA(r2,SubImg2)
    PUSH(r2)

    or r2,r0,r0      ; Esto vacia r2 para visualizarlo mejor cuando entra en la
                    ; subrutina y ver que se carga bien el dato desde la pila.
    bsr ValorPixel

Test3VPixel:
    LEA(r2,MFiltro3)
    PUSH(r2)
    LEA(r2,SubImg3)
    PUSH(r2)

    or r2,r0,r0      ; Esto vacia r2 para visualizarlo mejor cuando entra en la
                    ; subrutina y ver que se carga bien el dato desde la pila.
    bsr ValorPixel

Test4VPixel:
    LEA(r2,MFiltro4)
    PUSH(r2)
    LEA(r2,SubImg4)
    PUSH(r2)

    or r2,r0,r0      ; Esto vacia r2 para visualizarlo mejor cuando entra en la
                    ; subrutina y ver que se carga bien el dato desde la pila.
    bsr ValorPixel

```

```

;*****
;Matrices para SubMatriz
;*****
; SubMatriz ( Imagen, SubImg, i ,j)
;
;
; SubMaImagen1 -->| 1 2 3 |
;                | 4 5 6 |
;                | 7 8 9 |
;
; SubMaSubImg1 -->| 0 0 0 |
;                | 0 0 0 |
;                | 0 0 0 |
;
; i y j van desde 0 hasta X

SubMaImagen1: data 3,3,0x04030201,0x08070605,0x00000009
SubMaSubImg1: data 0,0,0
               ,0,0,0
               ,0,0,0

; La posicion [1,2] corresponde al 6

i1: data 1
j1: data 1

;-----;

; SubMaImagen2 -->| 1 2 3 4 5 |
;                | 2 6 7 8 4 |
;                | 3 7 9 7 3 |
;                | 4 8 7 6 2 |
;                | 5 4 3 2 1 |
;
;
; SubMaSubImg2 -->| 1 1 1 |
;                | 1 0 1 |
;                | 1 1 1 |
;
SubMaImagen2: data 5,5,0x04030201,0x07060205,0x07030408,0x04030709,0x02060708,0x02030405,0x00000001
SubMaSubImg2: data 1,1,1
               ,1,0,1
               ,1,1,1

; La posicion [2,2] corresponde al 9

i2: data 2
j2: data 2

;-----;

; SubMaImagen3 -->| 1 2 3 4 5 |
;                | 2 6 7 8 4 |
;                | 3 7 9 7 3 |
;                | 4 8 7 6 2 |
;                | 5 4 3 2 1 |
;
;
; SubMaSubImg3 -->| 1 1 1 |
;                | 1 0 1 |
;                | 1 1 1 |
;
SubMaImagen3: data 5,5,0x04030201,0x07060205,0x07030408,0x04030709,0x02060708,0x02030405,0x00000001
SubMaSubImg3: data 1,1,1
               ,1,0,1
               ,1,1,1

; La posicion [3,3] corresponde al 9

i3: data 3
j3: data 3

;-----;

; SubMaImagen4 -->| 10 20 30 |
;                | 40 50 60 |
;                | 70 80 90 |
;
;
; SubMaSubImg4 -->| -1 -1 -1 |
;                | -1 -1 -1 |
;                | -1 -1 -1 |
;
SubMaImagen4: data 3,3, 0x40302010, 0x08070605, 0x90
SubMaSubImg4: data 0xFFFFFFFF,0xFFFFFFFF,0xFFFFFFFF

; La posicion [1,1] corresponde al 50

i4: data 1
j4: data 1

; Resultado: Dir.Memoria --> 10203040 50607080 90FFFFFF

```

```

;-----;
SubMeImagen5 --> 01 02 03 04 05
                06 07 08 09 0A
                0B 0C 0D 0E 0F
                10 11 12 13 14
                15 16 17 18 19

SubMeSubIng5 --> -1 -1 -1
                -1 -1 -1
                -1 -1 -1

SubMeImagen5: data 5,5, 0x04030201, 0x08070605, 0x0C0B0A09, 0x100F0E0D, 0x14131211, 0x18171615, 0x1C
SubMeSubIng5: data 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF

; La posicion [4,4] corresponde al 19

15:      data 4
16:      data 4

; Resultado: Dir.Memoria --> 19191919 19191919 19FFFFFF

;-----;

SubMeImagen6 --> 1 2 3 4 5 6 7 8
                9 10 11 12 13 14 15 16
                17 18 19 20 21 22 23 24
                25 26 27 28 29 30 31 32
                33 34 35 36 37 38 39 40

SubMeSubIng6 --> -1 -1 -1
                -1 -1 -1
                -1 -1 -1

SubMeImagen6: data 5,0, 0x04030201, 0x08070605, 0x12111009, 0x16151413, 0x20191817, 0x24232221, 0x28272625
                , 0x32313029, 0x36353433, 0x40393837
SubMeSubIng6: data 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF

; La posicion [4,0] corresponde al 35

16:      data 4
17:      data 0

; Resultado: Dir.Memoria --> 35353535 35353535 35FFFFFF

;-----;

SubMeImagen7 --> 1 2 3 4 5 6 7 8
                9 10 11 12 13 14 15 16
                17 18 19 20 21 22 23 24
                25 26 27 28 29 30 31 32
                33 34 35 36 37 38 39 40

SubMeSubIng7 --> -1 -1 -1
                -1 -1 -1
                -1 -1 -1

SubMeImagen7: data 5,0, 0x04030201, 0x08070605, 0x12111009, 0x16151413, 0x20191817, 0x24232221, 0x28272625
                , 0x32313029, 0x36353433, 0x40393837
SubMeSubIng7: data 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF

; La posicion [3,4] corresponde al 30

17:      data 3
17:      data 5

; Resultado: Dir.Memoria --> 21222229 30313738 39FFFFFF

;-----;

SubMeImagen8 --> 1 2 3 4 5 6 7 8
                9 10 11 12 13 14 15 16
                17 18 19 20 21 22 23 24
                25 26 27 28 29 30 31 32
                33 34 35 36 37 38 39 40

SubMeSubIng8 --> -1 -1 -1
                -1 -1 -1
                -1 -1 -1

SubMeImagen8: data 5,8, 0x04030201, 0x08070605, 0x12111009, 0x16151413, 0x20191817, 0x24232221, 0x28272625
                , 0x32313029, 0x36353433, 0x40393837
SubMeSubIng8: data 0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF

; La posicion [4,7] corresponde al 40

18:      data 4
18:      data 7

; Resultado: Dir.Memoria --> 40404040 40404040 40FFFFFF

```

```

; Pruebas para SubMatriz
; SubMatriz ( Imagen, SubImg, i ,j)

Test1SubMa
    LOAD(r2,j1)
    PUSH(r2)
    LOAD(r2,i1)
    PUSH(r2)
    LEA(r2,SubMaSubImg1)
    PUSH(r2)
    LEA(r2,SubMaImagen1)
    PUSH(r2)
    OR r2,r0,r0 ; Esto vacia r2 para visualizarlo mejor cuando entra en la
                ; subrutina y ver que se carga bien el dato desde la pila.
    bsr SubMatriz

Test2SubMa
    LOAD(r2,j2)
    PUSH(r2)
    LOAD(r2,i2)
    PUSH(r2)
    LEA(r2,SubMaSubImg2)
    PUSH(r2)
    LEA(r2,SubMaImagen2)
    PUSH(r2)
    OR r2,r0,r0 ; Esto vacia r2 para visualizarlo mejor cuando entra en la
                ; subrutina y ver que se carga bien el dato desde la pila.
    bsr SubMatriz

Test3SubMa
    LOAD(r2,j3)
    PUSH(r2)
    LOAD(r2,i3)
    PUSH(r2)
    LEA(r2,SubMaSubImg3)
    PUSH(r2)
    LEA(r2,SubMaImagen3)
    PUSH(r2)
    OR r2,r0,r0 ; Esto vacia r2 para visualizarlo mejor cuando entra en la
                ; subrutina y ver que se carga bien el dato desde la pila.
    bsr SubMatriz

Test4SubMa
    LOAD(r2,j4)
    PUSH(r2)
    LOAD(r2,i4)
    PUSH(r2)
    LEA(r2,SubMaSubImg4)
    PUSH(r2)
    LEA(r2,SubMaImagen4)
    PUSH(r2)
    OR r2,r0,r0 ; Esto vacia r2 para visualizarlo mejor cuando entra en la
                ; subrutina y ver que se carga bien el dato desde la pila.
    bsr SubMatriz

```

```

Test5SubMa
    LOAD(r2,j5)
    PUSH(r2)
    LOAD(r2,i5)
    PUSH(r2)
    LEA(r2,SubMaSubImg5)
    PUSH(r2)
    LEA(r2,SubMaImagen5)
    PUSH(r2)
    OR r2,r0,r0 ; Esto vacia r2 para visualizarlo mejor cuando entra en la
                ; subrutina y ver que se carga bien el dato desde la pila.
    bsr SubMatriz

Test6SubMa
    LOAD(r2,j6)
    PUSH(r2)
    LOAD(r2,i6)
    PUSH(r2)
    LEA(r2,SubMaSubImg6)
    PUSH(r2)
    LEA(r2,SubMaImagen6)
    PUSH(r2)
    OR r2,r0,r0 ; Esto vacia r2 para visualizarlo mejor cuando entra en la
                ; subrutina y ver que se carga bien el dato desde la pila.
    bsr SubMatriz

Test7SubMa
    LOAD(r2,j7)
    PUSH(r2)
    LOAD(r2,i7)
    PUSH(r2)
    LEA(r2,SubMaSubImg7)
    PUSH(r2)
    LEA(r2,SubMaImagen7)
    PUSH(r2)
    OR r2,r0,r0 ; Esto vacia r2 para visualizarlo mejor cuando entra en la
                ; subrutina y ver que se carga bien el dato desde la pila.
    bsr SubMatriz

Test8SubMa
    LOAD(r2,j8)
    PUSH(r2)
    LOAD(r2,i8)
    PUSH(r2)
    LEA(r2,SubMaSubImg8)
    PUSH(r2)
    LEA(r2,SubMaImagen8)
    PUSH(r2)
    OR r2,r0,r0 ; Esto vacia r2 para visualizarlo mejor cuando entra en la
                ; subrutina y ver que se carga bien el dato desde la pila.
    bsr SubMatriz

```



```

;-----
;datos para FilPixel
;-----
; WPixel = FilPixel (Imagen, i, j, MFiltro)

FilPImagen1: data 0,0,0x44332211,0x88776655,0x00000000
FilPMFiltro1: data 1,1,1,
               1,0,1,
               1,1,1

; La posicion [1,2] corresponde al 6

FilPi1: data 1
FilPj1: data 2

;-----
;

FilPImagen2: data 0,0,0x44332211,0x00000000,0x11223344,0x11223344,
               0x33445566,0x44332211,0x00000000
FilPMFiltro2: data 0,0,0,
               0,1,0,
               0,0,0

; La posicion [2,2] corresponde al 26

FilPi2: data 2
FilPj2: data 3

;-----
;

FilPImagen3: data 0,0,0x00000000,0x00000000,0x33445566,0x00000000,
               0x44556677,0x00000000,0x00000000,0x00000000
FilPMFiltro3: data 1,1,1,
               1,0,1,
               1,1,1

; La posicion [3,2] corresponde al 88

FilPi3: data 2
FilPj3: data 2

;-----
;

FilPImagen4: data 0,0,0x00112233,0x44556677,0x8899AABB,0x44556677,
               0x00000000,0x44556677,0x8899AABB,0x44556677
FilPMFiltro4: data 1,1,1,
               1,-0,1,
               1,1,1

; La posicion [2,2] corresponde al 4A = 74

FilPi4: data 1
FilPj4: data 1

;-----
;

FilPImagen5: data 0,0,0x40302010,0x00706050,0x44332211,0x88776655,
               0x44556677,0x44556677,0x30607080,0x10203040
FilPMFiltro5: data 1,0,1,
               2,1,2,
               1,0,1

; La posicion [1,1] corresponde al 28

FilPi5: data 1
FilPj5: data 1

```

```

FilPImagen6: data 4,8,0x40302010,0x80706050,0x44332211,0x88776655,
               0x44556677,0x44556677,0x50607080,0x10203040

FilPMFiltro6: data 2,2,2,
               2,2,2,
               2,2,2

; La posicion [3,0] corresponde al 80

FilPi6: data 3
FilPj6: data 0

; Resultado debe ser --> r29 = 0x900 == 2304
; El resultado se debe ajustar a 255

;-----
;

FilPImagen7: data 4,8,0x40302010,0xFF706050,0x44332211,0x88776655,
               0x44556677,0x44556677,0x506070FF,0x10203040

FilPMFiltro7: data 1,0,1,
               1,0,1,
               1,0,1

; La posicion [3,0] corresponde al FF

FilPi7: data 3
FilPj7: data 0

; Resultado debe ser --> r29 = 0x720 == 1824
; El resultado se debe ajustar a 255

```

```

; Pruebas para FillPixel
;
; vPixel = FillPixel (Imagen, i, j, HFilter)
;
TestFillPixel:
    LEA(r2, FillWFilter0)
    PUSH(r2)
    LOAD(r2, FillP1)
    PUSH(r2)
    LOAD(r2, FillP1)
    PUSH(r2)
    LEA(r2, FillPImage1)
    PUSH(r2)

    Bsr FillPixel

Test2FillPixel:
    LEA(r2, FillWFilter2)
    PUSH(r2)
    LOAD(r2, FillP2)
    PUSH(r2)
    LOAD(r2, FillP2)
    PUSH(r2)
    LEA(r2, FillPImage2)
    PUSH(r2)

    Bsr FillPixel

Test3FillPixel:
    LEA(r2, FillWFilter3)
    PUSH(r2)
    LOAD(r2, FillP3)
    PUSH(r2)
    LOAD(r2, FillP3)
    PUSH(r2)
    LEA(r2, FillPImage3)
    PUSH(r2)

    Bsr FillPixel

Test4FillPixel:
    LEA(r2, FillWFilter4)
    PUSH(r2)
    LOAD(r2, FillP4)
    PUSH(r2)
    LOAD(r2, FillP4)
    PUSH(r2)
    LEA(r2, FillPImage4)
    PUSH(r2)

    Bsr FillPixel

Test5FillPixel:
    LEA(r2, FillWFilter5)
    PUSH(r2)
    LOAD(r2, FillP5)
    PUSH(r2)
    LOAD(r2, FillP5)
    PUSH(r2)
    LEA(r2, FillPImage5)
    PUSH(r2)

    Bsr FillPixel

Test6FillPixel:
    LEA(r2, FillWFilter6)
    PUSH(r2)
    LOAD(r2, FillP6)
    PUSH(r2)
    LOAD(r2, FillP6)
    PUSH(r2)
    LEA(r2, FillPImage6)
    PUSH(r2)

    Bsr FillPixel

Test7FillPixel:
    LEA(r2, FillWFilter7)
    PUSH(r2)
    LOAD(r2, FillP7)
    PUSH(r2)
    LOAD(r2, FillP7)
    PUSH(r2)
    LEA(r2, FillPImage7)
    PUSH(r2)

    Bsr FillPixel

```

```

;*****
; Pruebas para Filtro
;*****
; Filtro (Imagen, ImFiltrada, MFiltro)

Test1Filtro:
    LEA(r2,filtroFiltro1)
    PUSH(r2)
    LEA(r2,filtroFilt1)
    PUSH(r2)
    LEA(r2,filtroImagen1)
    PUSH(r2)

    bsr Filtro

Test2Filtro:
    LEA(r2,filtroFiltro2)
    PUSH(r2)
    LEA(r2,filtroFilt2)
    PUSH(r2)
    LEA(r2,filtroImagen2)
    PUSH(r2)

    bsr Filtro

Test3Filtro:
    LEA(r2,filtroFiltro3)
    PUSH(r2)
    LEA(r2,filtroFilt3)
    PUSH(r2)
    LEA(r2,filtroImagen3)
    PUSH(r2)

    bsr Filtro

```

```

;*****
; Datos para Filtro
;*****
; Filtro (Imagen, ImFiltrada, MFiltro)

filtroImagen1: data 4,8,0x04030201,0x07060504,0x14134211,0x17168514,
                  0x24232221,0x27262574,0x34333231,0x37363534

filtroFilt1: res 40

filtroFiltro1: data 0,-3,0,
                  0,4,0,
                  0,0,0

;-----;

filtroImagen2: data 4,6,0x04030201,0x02000605,0x05030104,
                  0x0C090603,0x0804120F,0x80402010

filtroFilt2: data 0xA5A5A5A5,0xA5A5A5A5,0xA5A5A5A5,0xA5A5A5A5,
                 0xA5A5A5A5,0xA5A5A5A5,0xA5A5A5A5,0xA5A5A5A5

filtroFiltro2: data -2,0,-2,
                  0,0,0,
                  -2,0,-2

;-----;

filtroImagen3: data 4,6,0x12345678,0xFFFFEFD0,0x13355779,
                  0xEEEEDECEB,0x23456789,0xDFDEDDDC

filtroFilt3: data 0x00000000,0x00000000,0x01020304,0x05060102,
                 0x03040506,0x01020304,0x05060102,0x03040506

filtroFiltro3: data 0,0,0,
                  0,1,0,
                  0,0,0

```

```

;*****
;Datos para FiltRec
;*****
; Diferencia = FiltRec (ImagenIn, ImagenOut, MFiltro, ModMFiltro, NCambios)

filtRecimg1:
    data 4,4,0x04030201,0x0D0E0F10,0x05040302,0x23222120
filtRecModM1:
    data 1,1
filtRecFiltro1:
    data 1,1,1,
        1,0,1,
        1,1,1
filtRfiltra1: res 24
    data 0xAAAAAAAA,0xAAAAAAAA

;-----;

filtRecimg2:
    data 4,4,0x04030201,0x0D0E0F10,0x05040302,0x23222120
filtRecModM2:
    data 1,2
filtRecFiltro2:
    data 10,10,10,
        10,00,10,
        10,10,10
filtRfiltra2: res 24
    data 0xAAAAAAAA,0xAAAAAAAA

```

```

;*****;
; Pruebas para FiltRec
;*****;
; Diferencia = FiltRec (ImagenIn, ImagenOut, MFiltro, ModMFiltro, NCambios)

Test1FiltRec:

    or r2,r0,r0
    or r3,r0,r0

    addu r2,r2,40      ; NCambios = 40
    addu r3,r3,4       ; nF = 4
    st r3,r0,0         ; Se carga nF en 0

    PUSH(r2)           ; Se carga NCambios en la pila
    LEA(r20,filtRecModM1)
    PUSH(r20)           ; Se carga ModMFiltro
    LEA(r20,filtRecFiltro1)
    PUSH(r20)
    LEA(r20,filtRfiltra1) ; ImagenOut
    PUSH(r20)
    LEA(r20,filtRecimg1)
    PUSH(r20)

    bsr FiltRec

Test2FiltRec:

    or r2,r0,r0
    or r3,r0,r0

    addu r2,r2,0       ; NCambios = 0
    addu r3,r3,4       ; nF = 4
    st r3,r0,0         ; Se carga nF en 0

    PUSH(r2)           ; Se carga NCambios en la pila
    LEA(r20,filtRecModM2)
    PUSH(r20)           ; Se carga ModMFiltro
    LEA(r20,filtRecFiltro2)
    PUSH(r20)
    LEA(r20,filtRfiltra2) ; ImagenOut
    PUSH(r20)
    LEA(r20,filtRecimg2)
    PUSH(r20)

    bsr FiltRec
    stop

```

3.Observaciones finales

La realización del proyecto ha tenido sus momentos divertidos, pero también los frustrantes. Me centraré en la parte frustrante, que no tiene nada que ver con haber realizado el proyecto individualmente, sino mas bien relacionado con el lenguaje y la hora de depurador.

Incluso al cuando ya no me quedaba demasiado para completarlo, no explote demasiado el simulador usando todas las posibilidades que este daba.

El proyecto cuando se inicia asusta un poco, pero supongo que es la sensación es la misma cuando se empieza a realizar cualquier proyecto o código, no saber por donde empezar. Además, para la época en la que inicie el proyecto estaba cerca el 1º parcial sobre la programación en ensamblador y no estaba seguro todavía de cómo funcionaba todavía. (Que este examen no salió demasiado bien)

Aunque en un principio la sensación del proyecto de que es muy difícil y creía que iba a necesitar mucho más tiempo, según avanzaba por él, esa sensación desapareció. El proyecto solo requiere de tiempo, paciencia y de no dejarlo para el final.

Ambos hitos fueron superados. Para el 1º de ellos fui muy apurado de tiempo y además, no pude aprovechar la corrección del 24 de octubre.

Para el 2º, no hubo problemas de tiempo, ya que fui mas consciente y pude entregar el código con mas rutinas que las requeridas para la superación del 2º hito.

La primera subrutina que tuve que depurar bastante fue "SubMatriz". Se podría decir que fue la primera que me dio verdaderos problemas. Para depurarla hice una batería de pruebas más amplia que la suministrada por la asignatura, con casos realizados a mano para luego conocer si se realizaban correctamente. Los problemas de esta subrutina venían por el manejo de las matrices.

"SubMatriz" fue la 1º rutina en plantearme problemas, pero todavía me quedaba por ver las 3 ultimas.

"FilPixel" me planteo dudas sobre las subrutinas que usaba, incluso cuando pasaron pruebas. En "FilPixel" es cuando empiezo a utilizar correctamente los puntos de dirección de retorno y la pila. además, profundizo mas y empiezo a entender muchas cosas que hasta este punto, confiaban simplemente en que funcionaran. En este punto es cuando corrijo las subrutinas anteriores y reviso que todas las llamadas a otras subrutinas siempre retornen al llamante.

"Filtro" y "FiltRec" fue sencilla su construcción y su depuración comparada con "FilPixel" y "SubMatriz", supongo que llegado a este punto ya tenía mas experiencia y veía con mas rapidez como hacer el código de Filtro y "FiltRec" y como depurarlo.

En este punto, terminando la realización de la memoria para enviarle, me encuentro con el código subido, pero sin la memoria entregada, aun teniendo 1 corrección todavía por usar, y aunque el código está bien, perder una corrección solo por subir la memoria no parece estar bien. además, de que el gestor no me permite subir la memoria por separado.

4. Anexo

4.1 Pruebas a mano

Fecha: 16/11/2018

Asignatura: 88110

Pag: 22

Calculo de la memoria

$$3 \times 3 = 9 \quad \begin{array}{r} 4 \\ 2 \end{array}$$

$$1 \times 1 = 1 \quad \begin{array}{r} 4 \\ 10 \quad 0'2 \\ 2 \end{array}$$

$$2 \times 2 = 4 \quad \begin{array}{r} 4 \\ 0 \quad 1 \end{array}$$

$$5 \times 8 = 40 \quad \begin{array}{r} 4 \\ 00 \quad 10 \end{array}$$

$$3 \times 8 = 24 \quad \begin{array}{r} 4 \\ 0 \quad 6 \end{array}$$

$$2 \times 7 = 14 \quad \begin{array}{r} 4 \\ 2 \quad 3+1 \end{array}$$

$$1B = 8 \text{ bits}$$

$$2 \times 7 = \begin{array}{r} \cancel{8 \times 2} \quad \cancel{8 \times 6} \quad \cancel{8 \times 2} \quad \cancel{8 \times 0} \quad \cancel{8 \times 4} \quad \cancel{8 \times 0} \quad \cancel{8 \times 0} \quad \cancel{8 \times 2} \\ 0 \times 01020304, \quad 0 \times 05060708, \\ 0 \times 09101112, \quad 0 \times 13141516 \end{array}$$

$$3 \cdot 4 = 12$$

$$(3+1) \cdot 4 = 16$$

$$(6+1) \cdot 4 = 28$$

2018

FiltRec

- r2 = M / Difusión de Comp
- r3 = N
- r4 = Total de comp/cantador
- r5 = Operación de memoria
- r6 = operación
- r7 = compara

- r21 = Imagen In
- r22 = Imagen Out
- r23 = M.Filtro
- r24 = Memoria Mod M.Filtro / Copia Imagen THP



Sublinea FiltRec : dir = 2060

Puntos = Imagen In = ~~0x2B78~~

It Imagen Out = ~~0x2B8C~~

bsr filtro, Dir = 2168

Sig Dir = 2172

Fuente: $4 \cdot 4 = 16$.

$$\begin{array}{r} 16 \ 4 \\ 0 \ 4 \end{array} + 1 = 5 \cdot 4$$

$$44 = 86 \quad +$$

$$208$$

Instrucciones
movs

+4

+4

+4

11

12

FiltRec : dir = 2060

Filtro : dir = 2168

largo : 2172

Fil Pixel : 1540

largo : 1444

FilPixel: st.b. : 2000

Bucle-M : br 2032

bsr ActualizaFiltro : 2272

Buc-Almacenas: ~~2328~~ ~~2332~~ ~~2336~~ **2336**

SigPaso : dir = 2568

nFiltRec : 2420

ImagenIn : 0x2B78

Imagen Out : 0x2B8C

nFiltro : 0x2B98

ModFiltro : 0x2B90

ImagenTMP: 0xEA30

NCambios : 0xEA48.

1771.

COMP: 2316

11 12, 150, 24, 2560

bsr FiltRec: 2612

FiltRec : 2620

Fin Pixel: Dir = 1564.

Fin.Fin Pixel: 1800

Subtrahir = 1640.

Valor Pixel = 1692

Largo de las 2 submatrices $\rightarrow 1712$.

Submatrices: 0x EA3C

Imagen: 0x2B25.

Ejemplo: 5x5

	0	1	2	3	4
0	11	22	33	44	55
1	1	2	3	4	5
2	21	22	23	24	25
3	31	32	33	34	35
4	41	42	43	44	45

Voy a coger el $[3,2] = 33$.

1: Subtrahir

22	23	24
32	33	34
42	43	44

2: Atilho. $\begin{pmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{pmatrix}$

Resultado = 33

$$22 \cdot 0 + 23 \cdot 0 + 24 \cdot 0 + \dots$$

$$\dots + 33 \cdot 1 + \dots$$

$$= 33$$

La prueba que elige en las cosas
de prueba a $[3,3]$

Test 3 Filled

$61 \cdot 8 = 488$

Matrix: 4×8

	0	1	2	3	4	5	6	7
0	44	44	44	44	44	44	44	44
1	44	34	34	33	44	44	44	44
2	44	44	88	44	44	44	44	44
3	44	44	44	44	44	44	44	44

Matrix: $\begin{pmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$

	0	1	2	3	4	5	6	7
0	X	X	X	X	X	X	X	X
1	34	34	34	X	X	X	X	X
2	X	88	X	X	X	X	X	X
3	X	X	X	X	X	X	X	X

Example 1: $[1, 1] = 34$

SubJug = $\begin{pmatrix} 44 & 44 & 44 \\ 44 & 34 & 34 \\ 44 & 44 & 44 \end{pmatrix}$

Example 2: $[2, 2] = 88$

SubJug = $\begin{pmatrix} 34 & 34 & 33 \\ 44 & 88 & 44 \\ 44 & 44 & 44 \end{pmatrix}$

$44 \cdot 7 + 34 = 342$

$342 / 8 = 42 \text{ ' } 75 = \boxed{42}$

$34 \cdot 2 + 33 + 44 \cdot 5 = 321$

$321 / 8 = 40 \text{ ' } 125 = \boxed{40}$

Example 3: $[1, 3] = 33$

SubJug = $\begin{pmatrix} 44 & 44 & 44 \\ 34 & 33 & 44 \\ 88 & 44 & 44 \end{pmatrix}$

Example 4: $[1, 2] = 54$

SubJug = $\begin{pmatrix} 44 & 44 & 44 \\ 34 & 34 & 33 \\ 44 & 88 & 44 \end{pmatrix}$

$44 \cdot 5 + 88 + 34 + 33 = 375$

$375 / 8 = 46 \text{ ' } 875$

$44 \cdot 6 + 34 + 88 = 386$

$386 / 8 = 48 \text{ ' } 2 = \boxed{48}$

$\boxed{46}$

Example 5: $[1, 6] = 44$

SubJug = $\begin{pmatrix} 44 & 44 & 44 \\ 44 & 44 & 44 \\ 44 & 44 & 44 \end{pmatrix}$

$44 \cdot 8 = 352$

$352 / 8 = 44$

Fecha: 11/1/2018

Asignatura: 88110 Pag: 25

$$\begin{array}{r} 488 \\ 08 \\ 08 \\ 0 \end{array} \begin{array}{r} 12 \\ 244 \\ 04 \\ 0 \end{array} \begin{array}{r} 12 \\ 176 \\ 02 \\ 0 \end{array} \begin{array}{r} 12 \\ 61 \\ 0 \end{array} \begin{array}{r} 12 \\ 61 \\ 0 \end{array}$$

$$\begin{array}{r} 488 \\ 244 \\ 122 \\ 61 \end{array} \begin{array}{r} 2 \\ 2 \\ 2 \end{array} \begin{array}{r} 4 \\ 8 \\ 8 \end{array}$$

61.8

~~488~~

~~08~~

61 es primo

~~08~~

Test 4 Fil/Row

Matrix = 4x8

	0	1	2	3	4	5	6	7
0	40	41	42	43	44	45	46	47
1	48	49	4A	4B	4C	4D	4E	4F
2	40	00	42	43	44	45	4C	47
3	48	49	4A	4B	4C	4D	4E	4F

$$\text{Matrix} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

func [5,2] = 47.

Totstkl. vel.

$$[2,3] = 44.$$

$$\begin{pmatrix} 34 & 33 & 44 \\ 88 & 44 & 44 \\ 44 & 44 & 44 \end{pmatrix} \quad \text{Hr.} \begin{pmatrix} 4 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

$$44_{16} = 68_{10}$$

$$68 \cdot 5 + 136 + 52 + 51 \\ = 579 / 8 = 72.$$

or

$$34_{16} = 52_{10}$$

$$33_{16} = 51_{10}$$

$$88_{16} = 136_{10}.$$

$$68 \cdot 7 + 52$$

$$\begin{pmatrix} 34 & 34 & 33 \\ 44 & 88 & 44 \\ 44 & 44 & 44 \end{pmatrix}$$

$$52 \cdot 2 + 51 + 68 \cdot 5 = 495 / 8 = 61'88$$

Fecha: 19/11/2018

Asignatura: 85110 Pag: 6

Til Pixel: ~~1648~~ 1564

Submáiz: 1648

KahPixel: 1692

Til - Til Pixel: 1800 (base de angua
r24 con el valor)

divs: 1766

Discusión de Puntos

r21 = EAC, ~~Submáiz~~

r22 = 2B78, Imagen

r23 = 2BA0, MFilter

Ejemplo $[2,2] = 85$.

$$1EF = 495/8 = 61$$

Test 4 til Fixel

hith = 498



$$V_{tilko} = \begin{pmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

	0	1	2	3	4	5	6	7
0	4B	42	41	40	47	46	45	44
1	4B	4A	49	48	4F	4E	4D	4C
2	4B	4L	00	40	47	46	45	44
3	4B	4A	49	48	4F	4E	4D	4C

Case $[z, 2] = 0$.

Case $[1, 1] = 4A$

$$\begin{pmatrix} 4A & 49 & 48 \\ 42 & 0 & 40 \\ 4A & 49 & 48 \end{pmatrix}$$

$$\begin{pmatrix} 43 & 42 & 41 \\ 4B & 4A & 49 \\ 43 & 42 & 0 \end{pmatrix} \begin{matrix} \bullet 66 \\ FE44 \\ 85 \end{matrix} \begin{matrix} 67+66+65+75+(-74.8) \\ 44+73+67+66 \\ = 479 - 592 = -113 \end{matrix}$$

$$\begin{aligned} & 67+66+65 \\ & + 75 + (-74.8) + 73 \\ & + 67 + 66 + 0. \end{aligned}$$

$$479 - 592 = -113$$

På den
0.

Test 5 Fil Rnd

Punto que falta: Matriz 4×8 . El filtro devuelve la suma promedio de los elementos de la columna y la fila correspondiente.

Matriz

	1	2	3	4	5	6	7	8
0	10	20	30	40	50	60	70	80
1	11	22	33	44	55	66	77	88
2	55	44	55	44	55	44	55	44
3	20	30	20	40	30	20	10	

$$\text{Matriz 1} \begin{pmatrix} 1 & 0 & -1 \\ 2 & 1 & -2 \\ 1 & 0 & -3 \end{pmatrix}$$

$$\text{Matriz 2} \begin{pmatrix} 4 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{pmatrix}$$

Caso: $[1,1] = 22$.

$$M: \begin{pmatrix} 10 & 20 & 30 \\ 11 & 22 & 33 \\ 55 & 44 & 55 \end{pmatrix}$$

Usando Matriz 1

$$16 \cdot 1 + 48(-1) + 17 \cdot 2 + 34$$

$$+ 51(-2) + 65 + 85(-3)$$

$$169 - 405 = -236$$

↓
Ajustar a 0

Caso $[1,4] = 55$

$$M: \begin{pmatrix} 40 & 50 & 60 \\ 44 & 55 & 66 \\ 44 & 55 & 44 \end{pmatrix}$$

Usando Matriz 1

Caso $[3,0] = 80$

$$M: \begin{pmatrix} 10 & 20 & 30 \\ 30 & 30 & 30 \\ 30 & 30 & 30 \end{pmatrix}$$

Matriz 2 usada

$$256 \cdot 9 = 2304$$

↓
Ajustar a 255

$$\text{Matriz 3:} \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

Si el filtro da un peso negativo.

Fecha: 18/11/2018

Asignatura: SS110

Pag: 28

Filtro: 1824

Isr Filter: 1952

stb \rightarrow ~~2008~~

Submarch \rightarrow 1648

Kalun Pixel \rightarrow

Fin Filter \rightarrow 2044

Caso 1

Imagen: 0x2D1C

Intelfitudo: 0x2D44

MFiltro: 0x2D6C

Caso 2:

Imagen: 0x2D90

Intelfitudo: 0x2DB0

MFiltro: 0x2DD0

Imagen: 0x2DF4

Intelfitudo: 0x2E18

MFiltro: 0x2E38

Test Filter:

Actuiz: 4x8

	0	1	2	3	4	5	6	7
0	1	2	3	4	5	6	7	
1	11	42	13	14	25	16	17	
2	21	22	23	24	25	26	27	
3	31	32	33	34	35	36	37	

$$\text{Filter: } \begin{pmatrix} 0 & -3 & 0 \\ 0 & 4 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

(1824)

[0,0] = 1

$$H = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

$$\rightarrow 1(-3) + 1 \cdot 4 = 1$$

[0,4] = 2

$$H = \begin{pmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{pmatrix}$$

$$\begin{array}{r} 2(-3) + 2 \cdot 4 = 2 \\ -6 \quad +8 \end{array}$$

[1,1] = 42

$$H = \begin{pmatrix} 1 & 2 & 3 \\ 11 & 42 & 13 \\ 21 & 22 & 23 \end{pmatrix}$$

$$2(-3) + 4 \cdot 42 = -6 + 168 = \underline{\underline{162}}$$

Test 2 Filtro

Matriz 4x6

	0	1	2	3	4	5
0	1	2	3	4	5	6
1	0	2	4	1	3	5
2	3	6	9	0	7	2
3	4	8	10	20	40	80

Filtro: $\begin{pmatrix} -2 & 0 & -2 \\ 0 & 0 & 0 \\ -2 & 0 & -2 \end{pmatrix}$

$[0,0] = 1.$

$$\begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \end{pmatrix} \rightarrow -2 + (-2) + (-2) + (-2) = -8$$

Se ajusta a 0

$[1,2] = 4.$

$$\begin{pmatrix} 2 & 3 & 4 \\ 2 & 4 & 1 \\ 6 & 9 & 0 \end{pmatrix} \rightarrow \begin{matrix} 2(-2) + 4(-2) + 6(-2) \\ -4 - 8 - 12 - 24 = -48 \end{matrix}$$

36

Se ajusta a 0.

O no, si -

$$-48 / -8 = 6.$$

Fecha: 15/11/2018

Asignatura: 89110

Pag: 29

$\text{filtRec} : 2068$

$\text{filtro} : 2176$

$\text{AlmacenFiltro} : 2280$

$\text{br Buc-Almacen} : 2372$

$\text{Sglaso} : 2376$

$\text{br Mtl} = 2428$

$\text{br FilRec} = 2524$

$\text{FIN} : 2664$

Caso 1

$r21 = \text{Imagen In} = 0x2E58$

$r22 = \text{Imagen Out} = 0x2E9C$

$r23 = \text{Mfiltro} = 0x2E78$

$r2 = M \neq$

$r3 = N \neq$

$r25 = \text{Imagen THP} = 0xEAB0$

Caso 2

$r21 = \text{In} = 0x2EBC$

$r22 = \text{Out} = 0x2F00$

$r23 = \text{Mfiltro} = 0x2EDC$

$r25 = \text{THP} =$

$\text{filtRec} : 2068$

$\text{a} : +32$

$\text{Puntif} : 2100$

$\text{br Mfiltros} : 2428$

$\text{br Corp} : 2524$

Fecha: 18/11/2018

Asignatura: 28110 Pag: 30

Caso 2 Filtro

Matriz 4×4

	0	1	2	3
0	1	2	3	4
1	10	F	E	D
2	2	3	4	5
3	20	21	22	23

Filtro:

$$\begin{pmatrix} A & A & A \\ A & 0 & A \\ A & A & A \end{pmatrix}$$

pero

$[0,0] = 1$

$1 \cdot 8 = \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$

$A \cdot 8 = 80/80 = 1$

$[0,1] = 2$

$\begin{pmatrix} 2 & 2 & 2 \\ 2 & 2 & 2 \\ 2 & 2 & 2 \end{pmatrix}$

$(2A) \cdot 8 = 160/80 = 2$