

**Proyecto de Programación en Ensamblador  
Estructura de Computadores  
Grado en Ingeniería Informática**

**Ejemplos de casos de prueba**

**Departamento de Arquitectura y Tecnología de Sistemas Informáticos**

2018-2019 (primer semestre)



## Ejemplos

Este documento complementa la sección “Ejemplos” del enunciado del proyecto de programación en ensamblador (pág. 21). Contiene varios ejemplos de casos de prueba para cada una de las subrutinas que componen el proyecto.

Estos ejemplos no constituyen un juego de ensayo completo. No sirven por sí solos para comprobar el correcto funcionamiento de las subrutinas ni para detectar de forma completa en qué situaciones presentan comportamientos erróneos. Sin embargo, se pueden seguir como guía para la elaboración de juegos de ensayo más completos.

En cada ejemplo se presenta el contenido de la memoria principal antes y después de la ejecución de una subrutina. Este contenido se muestra tal y como lo hace el simulador del 88110 usando el comando “V”. En concreto, se muestra lo siguiente:

- Los datos de prueba suministrados como parámetros a las subrutinas.
- El contenido de la pila al comienzo de su ejecución.
- Los resultados que producen.

En algunos casos también se describen los datos de prueba en lenguaje ensamblador.

Por último, en la página 19, se incluye el programa de prueba de un caso concreto para la subrutina Comp.

En este procesador el direccionamiento se hace a nivel de byte y se utiliza el formato *little-endian*. En consecuencia, cada una de las palabras representadas a continuación de la especificación de la dirección debe interpretarse como formada por 4 bytes con el orden que se muestra en el ejemplo siguiente:

Direcciones de memoria, tal como las representa el simulador:

60000	04050607	05010000
-------	----------	----------

Direcciones de memoria, tal como se deben interpretar:

60000	04
60001	05
60002	06
60003	07

60004	05
60005	01
60006	00
60007	00

Valor de las palabras almacenadas en las posiciones 60000 y 60004, tal como la interpreta el procesador:

60000	0x07060504 = 117.835.012
60004	0x00000105 = 261

## Actualización del núcleo de filtrado

### Caso 1. Llamada a ActualizaFiltro

Llama a 'ActualizaFiltro' pasándole una matriz de filtro formada por nueve elementos distintos y una dupla que modifica el filtro dividiendo a la mitad cada uno de sus elementos

```

    org      0x8000
Filtro: data 4, 5, 6
        data 7, 8, 9
        data 1, 2, 3
Dupla:  data 1, 2

```

```
r30=36856 (0x8FF8)
```

```
Direcciones de memoria:
```

36848			00800000	24800000
32768	04000000	05000000	06000000	07000000
32784	08000000	09000000	01000000	02000000
32800	03000000	01000000	02000000	

*Resultado:*

```
r30=36856 (0x8FF8)
```

```
Direcciones de memoria:
```

36848			00800000	24800000
32768	02000000	02000000	03000000	03000000
32784	04000000	04000000	00000000	01000000
32800	01000000	01000000	02000000	

**Caso 2.** Llamada a ActualizaFiltro

Llama a 'ActualizaFiltro'pasándole una matriz de filtro formada por nueve elementos distintos y una dupla que multiplica por dos cada uno de los elementos del filtro

```

        org      0x8000
Filtro: data    4, 5, 6
        data     7, 8, 9
        data     1, 2, 3
Dupla:  data     2, 1

r30=36856 (0x8FF8)
Direcciones de memoria:
36848                                00800000      24800000

32768      04000000      05000000      06000000      07000000
32784      08000000      09000000      01000000      02000000
32800      03000000      02000000      01000000

```

*Resultado:*

```

r30=36856 (0x8FF8)
Direcciones de memoria:
36848                                00800000      24800000

32768      08000000      0A000000      0C000000      0E000000
32784      10000000      12000000      02000000      04000000
32800      06000000      02000000      01000000

```

## Número de filtrados

### Caso 3. Llamada a nFiltrados

Llama a 'nFiltrados' pasándole un parámetro no nulo para iniciar la variable nF, que tiene un valor nulo.

r30=36860 (0x8FFC)

Direcciones de memoria:

00000          00000000

36848

0E000000

*Resultado:*

r30=36860 (0x8FFC) r29=14 (0xE)

Direcciones de memoria:

00000          0E000000

### Caso 4. Llamada a nFiltrados

Llama a 'nFiltrados' pasándole un parámetro negativo para decrementar la variable nF, que tiene un valor positivo.

r30=36860 (0x8FFC)

Direcciones de memoria:

00000          0E000000

36848

FFFFFFFF

*Resultado:*

r30=36860 (0x8FFC) r29=13 (0xD)

Direcciones de memoria:

00000          0D000000

## Compara dos imágenes

**Caso 5.** Llamada a `Comp` Llama a '`Comp`', pasándole dos imágenes de 4x8 elementos que difieren en uno solo de ellos.

```

    org 0x8000
IMAGEN1:
    data    4, 8
    data    0x00000000, 0x00000000
    data    0x00000000, 0x00002100
    data    0x00000000, 0x00000000
    data    0x00000000, 0x00000000
IMAGEN2:
    data    4, 8
    data    0x00000000, 0x00000000
    data    0x00000000, 0x00000000
    data    0x00000000, 0x00000000
    data    0x00000000, 0x00000000

r30=36856 (0x8FF8)
Direcciones de memoria:
36848                                00800000    28800000

32768    04000000    08000000    00000000    00000000
32784    00000000    00210000    00000000    00000000
32800    00000000    00000000

32800                                04000000    08000000
32816    00000000    00000000    00000000    00000000
32832    00000000    00000000    00000000    00000000

```

*Resultado:*

r30=36856 (0x8FF8) r29=34 (0x22)

**Caso 6.** Llamada a `Comp` Llama a '`Comp`', pasándole dos imágenes de 4x8 elementos en las que difieren todos sus elementos en una o en dos unidades.

```

r30=36856 (0x8FF8)
Direcciones de memoria:
36848                                00800000    28800000

32768    04000000    08000000    55FF55FF    55FF55FF
32784    FF55FF55    FF55FF55    55FF55FF    55FF55FF
32800    FF55FF55    FF55FF55

32800                                04000000    08000000
32816    54FE54FE    54FE54FE    FD57FD57    FD57FD57
32832    54FE54FE    54FE54FE    FD53FD53    FD53FD53

```

*Resultado:*

r30=36856 (0x8FF8) r29=2 (0x2)

## Extracción de submatriz

### Caso 7. Llamada a SubMatriz

Llama a 'SubMatriz', pasándole una imagen de 3x3 elementos de la que se ha de extraer la subimagen correspondiente al elemento central.

```

    org 0x8000
IMAGEN:
    data    3, 3
    data    0x40302010, 0x80706050, 0x90
    org 0x8040
SUBIMAGEN:
    data    0xFFFFFFFF, 0xFFFFFFFF, 0xFFFFFFFF

```

r30=36848 (0x8FF0)

Direcciones de memoria:

36848	00800000	40800000	01000000	01000000
32768	03000000	03000000	10203040	50607080
32784	90000000			
32832	FFFFFFFF	FFFFFFFF	FFFFFFFF	

*Resultado:*

r30=36848 (0x8FF0)

Direcciones de memoria:

32832	10203040	50607080	90FFFFFF
-------	----------	----------	----------

### Caso 8. Llamada a SubMatriz

Llama a 'SubMatriz', pasándole una imagen de 5x5 elementos de la que se ha de extraer la subimagen correspondiente a una esquina (inferior derecha).

r30=36848 (0x8FF0)

Direcciones de memoria:

36848	00800000	40800000	04000000	04000000
32768	05000000	05000000	01020304	05060708
32784	090A0B0C	0D0E0F10	11121314	15161718
32800	19000000			
32832	FFFFFFFF	FFFFFFFF	FFFFFFFF	

*Resultado:*

r30=36848 (0x8FF0)

Direcciones de memoria:

32832	19191919	19191919	19FFFFFF
-------	----------	----------	----------



**Valor del píxel filtrado****Caso 9.** Llamada a **ValorPixel**

Llama a 'ValorPixel', pasándole una subimagen nula excepto en su elemento central y un filtro identidad.

```

        org      0x8000
SUBIMAGEN:
        data     0x00000000, 0x00000055, 0x00
        org      0x8010
FILTRO: data     0, 0, 0
        data     0, 1, 0
        data     0, 0, 0

r30=36856 (0x8FF8)
Direcciones de memoria:
36848                                     00800000      10800000

32768      00000000      55000000      00000000

32784      00000000      00000000      00000000      00000000
32800      01000000      00000000      00000000      00000000
32816      00000000

```

*Resultado:*

r30=36856 (0x8FF8) r29=85 (0x55)

**Caso 10.** Llamada a **ValorPixel**

Llama a 'ValorPixel', pasándole una subimagen no nula y un filtro que dobla y cambia el signo del elemento al que se aplica.

```

r30=36856 (0x8FF8)
Direcciones de memoria:
36848                                     00800000      10800000

32768      00000000      55000000      00000000

32784      00000000      00000000      00000000      00000000
32800      FFFFFFFF      00000000      00000000      00000000
32816      00000000

```

*Resultado:*

r30=36856 (0x8FF8) r29=-170 (0xFFFFFFFF56)

**Caso 11.** Llamada a `ValorPixel`

Llama a 'ValorPixel', pasándole una subimagen no nula y un filtro que devuelve el valor negativo del doble de la suma de los ocho elementos que lo rodean. Los registros parten de valores distintos de 0.

r30=36856 (0x8FF8)

Direcciones de memoria:

36848			00800000	10800000
32768	10111213	14151617	18000000	
32784	FFFFFFFF	FFFFFFFF	FFFFFFFF	FFFFFFFF
32800	00000000	FFFFFFFF	FFFFFFFF	FFFFFFFF
32816	FFFFFFFF			

*Resultado:*

r30=36856 (0x8FF8) r29=-320 (0xFFFFFEC0)

**Filtro de un píxel****Caso 12. Llamada a FilPixel**

Llama a 'FilPixel', pasándole una imagen de 5x5 elementos y un filtro identidad que se aplica a un elemento del interior de la imagen.

```

        org      0x8000
IMAGEN:
        data     5, 5
        data     0x44332211, 0x03020155
        data     0x22210504, 0x31252423
        data     0x35343332, 0x44434241
        data     0x00000045

FILTRO: data     0, 0, 0
        data     0, 1, 0
        data     0, 0, 0

r30=36848 (0x8FF0)
Direcciones de memoria:
36848      00800000      02000000      03000000      24800000

32768      05000000      05000000      11223344      55010203
32784      04052122      23242531      32333435      41424344
32800      45000000

32800              00000000      00000000      00000000
32816      00000000      01000000      00000000      00000000
32832      00000000      00000000

```

*Resultado:*

```
r30=36848 (0x8FF0)  r29=36 (0x24)
```

**Caso 13.** Llamada a FilPixel

Llama a 'FilPixel', pasándole una imagen de 4x8 elementos y un filtro que devuelve la media de los ocho elementos que le rodean.

r30=36848 (0x8FF0)

Direcciones de memoria:

36848	00800000	02000000	02000000	30800000
32768	04000000	08000000	44444444	44444444
32784	44343433	44444444	44448844	44444444
32800	44444444	44444444		
32816	01000000	01000000	01000000	01000000
32832	00000000	01000000	01000000	01000000
32848	01000000			

*Resultado:*

r30=36848 (0x8FF0) r29=61 (0x3D)

**Caso 14.** Llamada a FilPixel

Llama a 'FilPixel', pasándole una imagen de 4x8 elementos y un filtro con peso cero que multiplica por -8 el valor del píxel y le suma el valor de los ocho píxeles que lo rodean. El resultado se ajusta al valor mínimo (0). Los registros parten de valores distintos de 0.

r30=36848 (0x8FF0)

Direcciones de memoria:

36848	00800000	02000000	02000000	28800000
32768	04000000	08000000	43424140	47464544
32784	4B4A4948	4F4E4D4C	43429940	47464544
32800	4B4A4948	4F4E4D4C		
32800			01000000	01000000
32816	01000000	01000000	F8FFFFFF	01000000
32832	01000000	01000000	01000000	

*Resultado:*

r30=36848 (0x8FF0) r29=0 (0x00000000)

## Filtro de imagen

### Caso 15. Llamada a Filtro

Llama a 'Filtro' pasándole una imagen no nula de 4x8 elementos y un filtro que multiplica por 4 cada elemento y le resta tres veces el valor del situado en la misma columna de la fila anterior. Algunos elementos alcanzan el valor máximo y otros el mínimo.

```

    org 0x8000
IMAGEN:
    data    4, 8
    data    0x04030201, 0x07060504
    data    0x14134211, 0x17168514
    data    0x24232221, 0x27262574
    data    0x34333231, 0x37363534
FILTRADA:
    res     40
FILTRO: data    0, -3, 0
        data    0,  4, 0
        data    0,  0, 0

r30=36852 (0x8FF4)
Direcciones de memoria:
36848                                00800000      28800000      50800000

32768      04000000      08000000      01020304      04050607
32784      11421314      14851617      21222324      74252627
32800      31323334      34353637

32800                                00000000      00000000
32816      00000000      00000000      00000000      00000000
32832      00000000      00000000      00000000      00000000

32848      00000000      FFFFFFFF      00000000      00000000
32864      04000000      00000000      00000000      00000000
32880      00000000

```

*Resultado:*

```

r30=36852 (0x8FF4)
Direcciones de memoria:
32800                                04000000      08000000
32816      01020304      04050607      11FF4344      44FF4617
32832      21005354      FF005627      31323334      34353637

```

**Caso 16.** Llamada a Filtro

Llama a 'Filtro'pasándole una imagen de 4x6 elementos y un filtro que sustituye cada píxel por la media de los que están situados en los cuatro vértices de la submatriz que lo rodea. El filtro utiliza coeficientes negativos.

r30=36852 (0x8FF4)

Direcciones de memoria:

36848		00800000	20800000	40800000
32768	04000000	06000000	01020304	05060002
32784	04010305	0306090C	0F120408	10204080
32800	A5A5A5A5	A5A5A5A5	A5A5A5A5	A5A5A5A5
32816	A5A5A5A5	A5A5A5A5	A5A5A5A5	A5A5A5A5
32832	FFFFFFFF	00000000	FFFFFFFF	00000000
32848	00000000	00000000	FFFFFFFF	00000000
32864	FFFFFFFF			

*Resultado:*

r30=36852 (0x8FF4)

Direcciones de memoria:

32800	04000000	06000000	01020304	05060004
32816	06080A05	03060A15	29120408	10204080

**Caso 17.** Llamada a Filtro Llama a 'Filtro'pasándole una imagen no nula de 4x6 elementos y un filtro identidad, que devuelve la misma imagen recibida.

r30=36852 (0x8FF4)

Direcciones de memoria:

36848		00800000	20800000	40800000
32768	04000000	06000000	78563412	FCFDFFEF
32784	79573513	EBECEDEE	89674523	DCDDDEDF
32800	00000000	00000000	04030201	02010605
32816	06050403	04030201	02010605	06050403
32832	00000000	00000000	00000000	00000000
32848	01000000	00000000	00000000	00000000
32864	00000000			

*Resultado:*

r30=36852 (0x8FF4)

Direcciones de memoria:

32800	04000000	06000000	78563412	FCFDFFEF
32816	79573513	EBECEDEE	89674523	DCDDDEDF

**Caso 18.** Llamada a `FiltRec` Llama a ‘FiltRec’ sobre una imagen de 4x4 elementos, con un filtro que devuelve para cada píxel la media de los que lo rodean. La dupla de actualización no modifica el filtro, el parámetro `NCambios` tiene valor 40 y `nF` se inicia a 4.

```
org 0x8000
IMAGEN:
    data      4, 4
    data      0x04030201, 0x0D0E0F10, 0x05040302, 0x23222120

DUPLA:  data   1, 1

FILTRO: data   1, 1, 1
       data   1, 0, 1
       data   1, 1, 1

FILTRADA: res    24
         data   0xAAAAAAAA, 0xAAAAAAAA

r30=36840 (0x8FEC)
Direcciones de memoria:
```

00000	04000000			
36832				00800000
36848	44800000	20800000	18800000	28000000
32768	04000000	04000000	01020304	100F0E0D
32784	02030405	20212223		
32784			01000000	01000000
32800	01000000	01000000	01000000	01000000
32816	00000000	01000000	01000000	01000000
32832	01000000			
32832		00000000	00000000	00000000
32848	00000000	00000000	00000000	
32848				AAAAAAAA
32864	AAAAAAAA			

*Resultado:*

r30=36840 (0x8FEC) r29=38 (0x26)

Direcciones de memoria

00000 03000000

36832 00800000

36848 44800000 20800000 18800000 28000000

32800 01000000 01000000 01000000 01000000

32816 00000000 01000000 01000000 01000000

32832 01000000

32832 04000000 04000000 01020304

32848 1005060D 02121305 20212223

32848 AAAAAAAAAA

32864 AAAAAAAAAA



**Caso 19.** Llamada a `FiltRec` llama a 'FiltRec' sobre una imagen de 4x4 elementos, con un filtro formado por coeficientes distintos de "1" que devuelve para cada píxel la media de los que lo rodean. La dupla de actualización modifica el filtro, el parámetro `NCambios` tiene valor 0 y `nF` se inicia a 4.

`r30=36840 (0x8FEC)`

Direcciones de memoria:

00000	04000000			
36832				00800000
36848	44800000	20800000	18800000	00000000
32768	04000000	04000000	01020304	100F0E0D
32784	02030405	20212223		
32784			01000000	02000000
32800	0A000000	0A000000	0A000000	0A000000
32816	00000000	0A000000	0A000000	0A000000
32832	0A000000			
32832		00000000	00000000	00000000
32848	00000000	00000000	00000000	
32848				AAAAAAAA
32864	AAAAAAAA			

*Resultado:*

`r30=36840 (0x8FEC) r29=-1 (0xFFFFFFFF)`

Direcciones de memoria

00000	00000000			
36832				00800000
36848	44800000	20800000	18800000	00000000
32800	00000000	00000000	00000000	00000000
32816	00000000	00000000	00000000	00000000
32832	00000000			
32832		04000000	04000000	01020304
32848	1008090D	02131305	20212223	
32848				AAAAAAAA
32864	AAAAAAAA			

**Caso 20.** Llamada a `FiltRec` Llama a 'FiltRec' sobre una imagen de 5x5 elementos, con un filtro que devuelve para cada píxel una media ponderada de los que lo rodean. La dupla de actualización decrementa levemente los valores del filtro, el parámetro `NCambios` tiene valor 1 y `nF` se inicia a 5.

`r30=36840 (0x8FEC)`

Direcciones de memoria:

00000	05000000			
36832				00800000
36848	50800000	2C800000	24800000	01000000
32768	05000000	05000000	0A000A00	0A000000
32784	00000A00	0A000A00	00000000	0A000A00
32800	0A000000			
32800		09000000	0A000000	
32800				00000000
32816	14000000	00000000	14000000	00000000
32832	14000000	00000000	14000000	00000000
32848	00000000	00000000	00000000	00000000
32864	00000000	00000000	00000000	00000000
32880	00000000			
32880		AAAAAAAA	AAAAAAAA	

*Resultado:*

`r30=36840 (0x8FEC) r29=0 (0x00)`

Direcciones de memoria

00000	01000000			
36832				00800000
36848	44800000	20800000	18800000	01000000
32800				00000000
32816	0C000000	00000000	0C000000	00000000
32832	0C000000	00000000	0C000000	00000000
32848	05000000	05000000	0A000A00	0A000203
32864	02000A03	04030A00	02030200	0A000A00
32880	0A000000			
32880		AAAAAAAA	AAAAAAAA	

```

;
;      Ejemplo de programa de prueba de la subrutina Comp
;

LEA:    MACRO    (ra, eti)
        or ra, r0, low(eti)
        or.u ra, ra, high(eti)
        ENDMACRO

PUSH:    MACRO(ra)
        subu r30, r30, 4
        st ra, r30, 0
        ENDMACRO

POP:     MACRO(ra)
        ld ra, r30, 0
        addu r30, r30, 4
        ENDMACRO

; Definición de la pila
        org      0xF000
PILA:    data     0

; Definición de las imágenes de prueba (2x2)
        org      0x1000
IMG1:    data     0x02, 0x02, 0x07050301

        org      0x2000
IMG2:    data     0x02, 0x02, 0x04030201

; Programa principal
        org      100
PPAL:    LEA (r30, PILA)      ; Inicialización del puntero de pila
        LEA (r1, IMG1)      ; r1: dirección de comienzo de IMG1
        LEA (r2, IMG2)      ; r2: dirección de comienzo de IMG2

        PUSH (r2)           ; Paso de parámetro Imagen2
        PUSH (r1)           ; Paso de parámetro Imagen1

        bsr Comp            ; Llamada a la subrutina Comp
                                ; r29 debe valer 3 (0x000000003)

        POP (r1)
        POP (r2)

        stop

Comp:    PUSH (r1)
;      ....

```