

Proyecto de Programación en Ensamblador Estructura de Computadores, Plan 2009

Departamento de Arquitectura y Tecnología de Sistemas Informáticos

Segundo Semestre, 2014-2015

Introducción

El proyecto consiste en la programación, en ensamblador del Motorola 88110, de un conjunto de rutinas que realicen el filtrado de una imagen mediante un filtro programable. La imagen será una matriz de píxeles, cada uno de los cuales se representa mediante un byte sin signo que especifica su nivel de gris (0 equivale a negro y 255 a blanco). El filtro está basado en una operación recursiva de convolución con un núcleo representado por una matriz constante, cuadrada de dimensión 3 y formada por números fraccionarios.

AVISO –2014/2015–

El enunciado de este proyecto **es diferente pero está basado en gran parte en el planteado para el curso anterior**. Los cambios introducidos responden a varios objetivos, entre los cuales está la idea de simplificar el desarrollo para los alumnos que se enfrentan por primera vez a este proyecto. Esto se ha realizado a la vez que se procuraba mantener la misma profundidad conceptual que tenía el proyecto en cursos anteriores. Para ello se ha procedido a descomponer alguna de las subrutinas más complejas en otras de menor entidad. Ello facilitará que el alumno realice una depuración más sencilla, basada en un mayor número de pasos más pequeños.

Debido a lo indicado en el párrafo anterior, aquellos alumnos que tengan que repetir o corregir el proyecto que desarrollaron en una convocatoria previa, podrán utilizar los programas ya realizados anteriormente, si bien *deberán construir y/o adaptar las diferentes subrutinas que se especifican para el curso actual, así como superar las pruebas que se establezcan para cada convocatoria*.

Por otra parte debe observar con atención el apartado de este documento que describe las normas de entrega y, en particular, la especificación del contenido que debe incluir la memoria del proyecto.

El hecho de plantear un proyecto que coincide en una buena parte con el del curso anterior tiene además las siguientes implicaciones:

- Se realizará una revisión minuciosa de los proyectos realizados en este semestre para descartar o localizar posibles **casos de copia** que desafortunadamente se siguen produciendo (y detectando) en la mayoría de las convocatorias.
- Los alumnos que ya hubieran formado parte de un grupo durante convocatorias anteriores **solo podrán establecer grupo con los mismos compañeros** que tuvieron en dicho curso o semestre o, alternativamente, realizar el proyecto **de forma individual**.

En el proyecto se programará en ensamblador una serie de subrutinas que permitan aplicar el filtro especificado en cada caso a la matriz que se proporcione como ejemplo, obteniendo como resultado una nueva matriz con la imagen filtrada. La función principal que debe realizar este conjunto de subrutinas se describe a continuación.

Transformar una imagen mediante la aplicación de un filtro genérico

El filtrado de una imagen consiste en realizar una serie de operaciones sobre cada uno de los píxeles que la componen. Puesto que se trabaja con imágenes en escala de grises, cada píxel corresponde a un byte sin signo, que indica su luminosidad. El filtro que se emplea en este proyecto pertenece a la categoría de los filtros lineales FIR, aunque se ha modificado para que su implementación permita profundizar en algunos conceptos importantes de la programación en ensamblador. El filtro se basa en la convolución discreta de dos matrices: la primera corresponde a la imagen a filtrar, y la segunda el propio filtro (también llamado el “núcleo” del filtro), que podría considerarse como otra imagen, pero de menor tamaño. De hecho, en el presente proyecto, este núcleo del filtro es una matriz de 3x3 elementos.

El modo de aplicación del filtro 3x3 consiste en efectuar, sobre cada píxel de la matriz que representa la imagen, ciertas operaciones en las que interviene el valor de dicho píxel, el de sus ocho vecinos más cercanos, y el de todos los elementos de la matriz que se toma como núcleo del filtro. Puesto que se trabaja con filtros cuyos valores numéricos no necesariamente son enteros, es posible que la aplicación de un filtro produzca como resultado un valor no entero. En estos casos, el valor obtenido se aproximará truncando. Además, puesto que los píxeles de las dos filas y de las dos columnas de los bordes de la imagen tienen un número de vecinos inferior a ocho, se aplicará la operación de filtrado de dichos píxeles de un modo particular.

Denominando $f_{i,j}$ al elemento situado en la fila i y en la columna j de la matriz que representa el núcleo del filtro, $m_{i,j}$ al elemento de la fila i y la columna j de la matriz que representa la imagen y $r_{i,j}$ al correspondiente elemento en la imagen filtrada, la operación de filtrado es la siguiente:

$$\begin{aligned} r_{i,j} = & f_{0,0} \times m_{i-1,j-1} + f_{0,1} \times m_{i-1,j} + f_{0,2} \times m_{i-1,j+1} + \\ & + f_{1,0} \times m_{i,j-1} + f_{1,1} \times m_{i,j} + f_{1,2} \times m_{i,j+1} + \\ & + f_{2,0} \times m_{i+1,j-1} + f_{2,1} \times m_{i+1,j} + f_{2,2} \times m_{i+1,j+1} \end{aligned}$$

Por ejemplo, en el caso representado en la Figura 1, el nuevo valor del píxel 2, 2 de la imagen $im_{original}$ ($m_{2,2} = 1$) tras aplicar el filtro $F1$ será $r_{2,2} = 4$:

$$\begin{aligned} r_{2,2} = & f_{0,0} \times m_{1,1} + f_{0,1} \times m_{1,2} + f_{0,2} \times m_{1,3} + f_{1,0} \times m_{2,1} + f_{1,1} \times m_{2,2} + f_{1,2} \times m_{2,3} + \\ & + f_{2,0} \times m_{3,1} + f_{2,1} \times m_{3,2} + f_{2,2} \times m_{3,3} = \\ = & 6 \times 0,125 + 7 \times 0,125 + 8 \times 0,125 + 0 \times 0,125 + 1 \times 0,0 + 2 \times 0,125 + \\ & + 4 \times 0,125 + 5 \times 0,125 + 6 \times 0,125 = 4 \quad (\text{tras el ajuste de } 4,75) \end{aligned}$$

En el caso de la implementación solicitada en este proyecto, el filtro anterior, especificado de forma genérica mediante una matriz cuadrada de coeficientes fraccionarios, se define en realidad como una terna de coeficientes constantes, tal como muestra el recuadro interno de la Figura 1. El primero de ellos, K , es una constante que divide a cada componente del filtro. El segundo, C , define el valor que, una vez dividido entre K , tendrá el elemento central del filtro (es decir, $f_{1,1} = C/K$). El último coeficiente, B , define el valor que, una vez dividido entre K , tendrá cada uno de los elementos del borde de la matriz (es decir, todos los elementos $f_{i,j}$ con la excepción del $f_{1,1}$ tendrán el valor $f_{i,j} = B/K$). En consecuencia, en el ámbito de este proyecto y dado que solo se puede especificar un único valor para los ocho elementos del borde de la matriz de filtro, todos los elementos de esta matriz, salvo posiblemente el central, serán idénticos.

Im_original ($m_{i,j}$)										Filtro F1 ($f_{i,j}$)		
1	2	3	4	5	6	7	.	.	.	$\begin{pmatrix} 0,125 & 0,125 & 0,125 \\ 0,125 & 0,0 & 0,125 \\ 0,125 & 0,125 & 0,125 \end{pmatrix}$	Definido por: K=8, C=0, B=1	
5	6	7	8	9	0	1	.	.	.			
9	0	1	2	3	4	5	.	.	.			
3	4	5	6	7	8	9	.	.	.			
7	8	9	0	1	2	3	.	.	.			
.	.	.	.	.	.	.	.	.	.			

Valor de $m_{2,2} = 1$
Valor de $r_{2,2} = 4,75 \rightarrow 4$

Figura 1. Aplicación de un filtro de imagen.

Según se señalaba anteriormente, el filtro de imagen presenta una modificación respecto al uso más frecuente de este tipo de filtros: consiste en que su aplicación será *recursiva*, de modo que se volverá a aplicar el mismo filtro sucesivas veces. De hecho, se aplicará de nuevo siempre que se cumplan las condiciones siguientes:

- En la última aplicación del filtro y de acuerdo a cierta medida, que es específica de este proyecto y se definirá más adelante, la imagen resultante difiere en un cierto grado de la imagen original.
- El número de veces que se ha aplicado el filtro es inferior a un máximo predeterminado (se definirá más adelante).

En la descripción de la rutina correspondiente a la aplicación del filtro quedarán aclarados los aspectos concretos de implementación.

Estructura del proyecto

El proyecto estará compuesto por seis subrutinas que se relacionan tal como se indica en la Figura 2. Además, el alumno o grupo construirá un programa principal para cada una de las pruebas que sea necesario efectuar durante la implementación y depuración de las subrutinas del proyecto. Se ha desglosado el programa de filtrado en un número alto de subrutinas para facilitar al alumno su depuración, ya que así se enfrentará a fragmentos de código de tamaño manejable. Además, al tener el programa segmentado en varias partes, el corrector automático proporcionará información más precisa sobre las partes que se han implementado correctamente y las que necesitan corregirse.

Programa Principal

El programa principal se encargará de inicializar la pila de usuario, almacenar en ella los parámetros que se deben pasar, e invocar a las distintas rutinas objetivo de este proyecto.

Los parámetros de las rutinas se pasarán siempre en la pila salvo que se especifique lo contrario. El resultado se recogerá normalmente en el registro *r29*, salvo que se especifique de otro modo.

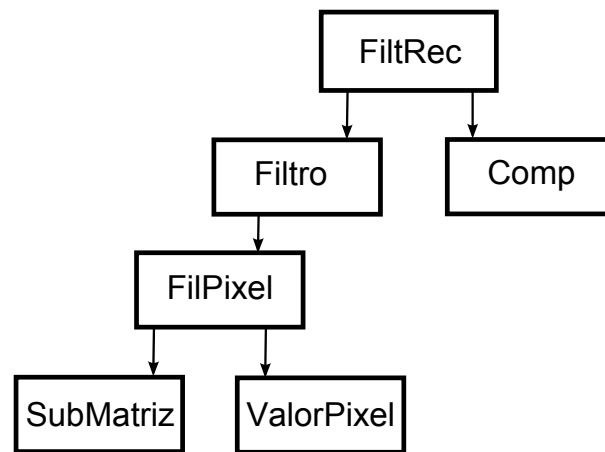


Figura 2. Jerarquía de las rutinas del proyecto.

Filtro recursivo

`NFiltrados = FiltRec ( Imagen, MFiltro, NCambios, NFiltrados )`

Parámetros:

- **Imagen:** Es la matriz que contiene tanto la imagen de entrada, a la que se ha de aplicar el filtro, como la imagen de salida, una vez filtrada. Se pasa por dirección, por lo que ocupa 4 bytes. Es un parámetro de entrada y salida que tiene tres campos:
 - El número de filas, de tipo entero (M).
 - El número de columnas, de tipo entero (N).
 - Los elementos almacenados por filas, cada uno de tipo byte sin signo (MxN bytes, MxN/4 palabras).
- **MFiltro.** Es la matriz cuadrada que define el filtro que se ha de aplicar a la imagen. Es un parámetro de entrada que se pasa por dirección, por lo que ocupa 4 bytes. Define la posición de memoria en que se encuentra la especificación del filtro, formada por tres constantes enteras (con signo) cada una de las cuales ocupa una palabra:
 - La constante K , que divide a cada componente del filtro.
 - El valor C del elemento central del filtro.
 - El valor B de cada uno de los ocho elementos pertenecientes al borde de la matriz de filtro.
- **NCambios:** Es el valor mínimo de la diferencia entre la imagen original y la filtrada que determina la necesidad de realizar un nuevo filtrado (una nueva llamada recursiva). Es un parámetro de entrada de tipo entero, se pasa por valor y ocupa 4 bytes. La diferencia entre una imagen y otra se especifica en la descripción de la subrutina `Comp`, ya que coincide con su valor de retorno.
- **NFiltrados:** Es un valor que define el número de veces que ya se ha aplicado a la imagen original el filtro definido por `MFiltro`. Es un parámetro de entrada, de tipo entero, se pasa por valor y ocupa 4 bytes.

Valor de retorno:

- **NFiltrados:** La función devuelve el número de veces que se ha aplicado el filtro a la imagen original. El valor de retorno se devuelve en el registro *r29*.

Descripción:

La rutina **FiltRec** realiza el filtrado **recursivo** de una imagen. La imagen de entrada está definida por el primer parámetro de la subrutina, que especifica el número de filas y columnas y los bytes que representan cada uno de sus píxeles. La labor básica de esta subrutina consiste en aplicar una función de filtrado a cada uno de los píxeles de la imagen (mediante la llamada a la subrutina **Filtro**) y comprobar si se ha alcanzado alguna de las condiciones de terminación de la recursividad, en cuyo caso se devuelve el control al llamante. La condición de salida de la recursividad es que se observe cualquiera de las siguientes situaciones:

- El número de veces que se ha aplicado el filtro ha alcanzado el máximo permitido (**15**). Puesto que este número de veces que ya se ha aplicado el filtro coincide con un parámetro de entrada, es necesario que el programa principal desde el que se llama a esta subrutina proporcione **0** como valor de este parámetro.
- En la última aplicación del filtro y de acuerdo a la medida proporcionada por la subrutina **Comp**, la diferencia entre la matriz sin filtrar y la matriz de entrada es menor al valor especificado en el parámetro *NCambios*.

La implementación de esta subrutina **debe realizarse obligatoriamente mediante el algoritmo recursivo aquí descrito**. En caso de entregar una implementación no recursiva, el proyecto será evaluado como suspenso aún en el caso de que supere todas las pruebas del corrector automático.

El funcionamiento de la rutina será el descrito a continuación:

1. Reservará espacio en el marco de pila para almacenar la imagen filtrada *ImFiltrada*. Lógicamente se reservará el mismo espacio que ocupa la imagen sin filtrar, pero ajustado por exceso para que sea un múltiplo de 4 bytes.
2. Llamará a la subrutina **Filtro**. Esta aplicará (una vez) el filtro definido por la matriz de coeficientes *MFiltro*. Para ello habrá preparado los parámetros de dicha llamada, que serán: la imagen original *Imagen*, la zona de memoria en que quedará la imagen filtrada (la zona que se ha reservado en el marco de pila en el paso anterior) *ImFiltrada*, y la matriz de coeficientes, *MFiltro*.
3. Incrementará la variable que mantiene el número de veces que ya se ha pasado el filtro, que pasará a ser $NFiltrados + 1$. Si $(NFiltrados + 1) \geq 15$, se abandonará el proceso recursivo (por requerir un número demasiado alto de aplicaciones del filtro). En este caso, se asignará al valor de retorno (registro *r29*) el valor actual de *NFiltrados* y la ejecución continuará en el paso 6 de este procedimiento.
4. Calculará la diferencia entre la imagen original y la filtrada para decidir si el proceso recursivo debe continuar o finalizar. Para ello, llamará a la subrutina **Comp**, pasándole

como parámetros las imágenes original y filtrada. Si el valor devuelto por `Comp` es menor que el especificado por el parámetro `NCambios`, se dará por terminado el proceso recursivo, para lo que se asignará al valor de retorno (registro `r29`) el valor actual de `NFiltrados` y la ejecución continuará en el paso 6 de este procedimiento.

5. Al no haberse alcanzado el final de la recursividad, llamará a la propia subrutina `FiltRec`, con los siguientes parámetros: la imagen filtrada, `ImFiltrada`, la matriz de coeficientes, `MFiltro`, el valor máximo de la diferencia entre imágenes para que se termine el proceso recursivo, `NCambios` y, finalmente, el número de pasadas del filtro ya realizadas, `NFiltrados`. Al retornar de esta llamada recursiva, la ejecución continuará por el paso 6.
6. Copiará la imagen filtrada `ImFiltrada` (la que ha quedado en el marco de pila) sobre la zona que ocupaba la imagen original, es decir, la zona determinada por el primer parámetro (`Imagen`).
7. Devolverá el control al programa llamante, proporcionándole como valor de retorno en `r29` el número de pasadas realizadas hasta el momento.

NOTA: Para facilitar la asignación correcta del valor de retorno evitando complicar innecesariamente la implementación de la subrutina, se aconseja no utilizar el registro `r29` en ningún fragmento de la segunda parte de la subrutina, la formada por las instrucciones comprendidas entre la llamada recursiva a la propia subrutina y la última (el retorno al llamante).

Filtro de imagen

`Filtro ( Imagen, ImFiltrada, MFiltro )`

Parámetros:

- **Imagen:** Es la matriz que contiene la imagen de entrada a la que se ha de aplicar el filtro. Se pasa por dirección, por lo que ocupa 4 bytes. Es un parámetro de entrada que tiene tres campos:
 - El número de filas, de tipo entero (M).
 - El número de columnas, de tipo entero (N).
 - Los elementos almacenados por filas, cada uno de tipo byte sin signo (MxN bytes, MxN/4 palabras).
- **ImFiltrada:** Es la matriz que contendrá el resultado de aplicar el filtro sobre la imagen de entrada. Se pasa por dirección, por lo que ocupa 4 bytes. Es un parámetro de salida que tiene tres campos:
 - El número de filas, de tipo entero (M).
 - El número de columnas, de tipo entero (N).
 - Los elementos almacenados por filas, cada uno de tipo byte sin signo (MxN bytes, MxN/4 palabras).

- **MFiltro.** Es la matriz cuadrada que define el filtro que se ha de aplicar a la imagen. Es un parámetro de entrada que se pasa por dirección, por lo que ocupa 4 bytes. Define la posición de memoria en que se encuentra la especificación del filtro, formada por tres constantes enteras (con signo) cada una de las cuales ocupa una palabra:
 - La constante K , que divide a cada componente del filtro.
 - El valor C del elemento central del filtro.
 - El valor B de cada uno de los ocho elementos pertenecientes al borde de la matriz de filtro.

Descripción:

La rutina **Filtro** aplica la máscara de filtrado definida por la matriz *MFiltro* a la imagen proporcionada como primer parámetro, *Imagen* y deja el resultado en la imagen especificada como segundo parámetro *ImFiltrada*.

El funcionamiento de esta rutina será el siguiente:

1. Copiará las constantes que definen el número de filas M y el número de columnas N de la imagen original sobre la imagen filtrada *ImFiltrada*.
2. Desde la primera ($i = 0$) hasta la última fila ($i = (M - 1)$), de la matriz original, realizará las siguientes operaciones:
 - a) Desde el primero ($j = 0$) al último ($j = (N - 1)$) píxel de la fila que se está procesando, se realizarán las siguientes operaciones:
 - Preparará los parámetros y realizará una llamada a la subrutina **FilPixel** para el elemento actual. Los parámetros que se proporcionan son los siguientes: la imagen original *Imagen*, el número de fila i , y de columna j , y el filtro, *MFiltro*.
 - Almacenará el valor de retorno recogido de la subrutina **FilPixel** en la posición que le corresponde (i, j) de la imagen filtrada *ImFiltrada*.

Filtro de un píxel

```
VPixel = FilPixel ( Imagen, i, j, MFiltro )
```

Parámetros:

- **Imagen:** Es la matriz que contiene la imagen de entrada a uno de cuyos píxeles se ha de aplicar el filtro. Se pasa por dirección, por lo que ocupa 4 bytes. Es un parámetro de entrada que tiene tres campos:
 - El número de filas, de tipo entero (M).
 - El número de columnas, de tipo entero (N).
 - Los elementos almacenados por filas, cada uno de tipo byte sin signo ($M \times N$ bytes, $M \times N / 4$ palabras).

- **i:** Es el número de fila (comprendido entre 0 y M-1) del píxel de la imagen proporcionada como primer parámetro *Imagen* al que se quiere aplicar la máscara definida por *MFiltro*. Es un parámetro de entrada que se pasa por valor.
- **j:** Es el número de columna (comprendido entre 0 y N-1) del píxel de la imagen proporcionada como primer parámetro *Imagen* al que se quiere aplicar la máscara definida por *MFiltro*. Es un parámetro de entrada que se pasa por valor.
- **MFiltro.** Es la matriz cuadrada que define el filtro que se ha de aplicar a la imagen. Es un parámetro de entrada que se pasa por dirección, por lo que ocupa 4 bytes. Define la posición de memoria en que se encuentra la especificación del filtro, formada por tres constantes enteras (con signo) cada una de las cuales ocupa una palabra:
 - La constante *K*, que divide a cada componente del filtro.
 - El valor *C* del elemento central del filtro.
 - El valor *B* de cada uno de los ocho elementos pertenecientes al borde de la matriz de filtro.

Valor de retorno:

- **VPixel:** La función devuelve el valor que se ha de asignar al píxel (*i, j*) de la imagen filtrada. El valor de retorno es un entero sin signo que se devuelve en el registro *r29*.

Descripción:

La rutina **FilPixel** aplica la máscara de filtrado definida por la matriz *MFiltro* al píxel seleccionado por los parámetros *i* y *j* de la imagen proporcionada como primer parámetro, *Imagen*. La rutina devuelve como resultado el valor equivalente al mismo píxel en la imagen filtrada. No será necesario comprobar que los valores de *i* y *j* facilitados como parámetros hacen referencia a un elemento válido de la imagen, sino que se supondrá que el programa o subrutina llamante solo proporcionará valores adecuados para dichos parámetros.

El funcionamiento de esta rutina será el siguiente:

1. Reservará espacio en el marco de pila para almacenar una submatriz cuadrada de tamaño 3×3 construida por la subrutina **SubMatriz** a partir del píxel que se está filtrando. Dado que la submatriz ocupará 9 bytes, el espacio que se reservará será de 3 palabras (12 bytes), ya que es el menor número de palabras en que se pueden almacenar los 9 bytes de la submatriz.
2. Llamará a la subrutina **SubMatriz**. Esta obtendrá y dejará almacenada en el espacio reservado en el paso anterior la submatriz 3×3 *SubImg* de la que tras aplicar el filtro definido por *MFiltro* se obtendrá el nuevo valor del píxel que se está filtrando. Para realizar esta llamada se habrán preparado sus parámetros: la imagen original *Imagen*, la zona de memoria en que quedará la submatriz resultado (la zona que se ha reservado en el marco de pila en el paso anterior) y los valores de *i* y *j*.
3. Llamará a la subrutina **ValorPixel**. Esta obtendrá el valor numérico del píxel una vez que se ha completado su filtrado. Para realizar esta llamada se habrán preparado

sus parámetros: la submatriz generada en el paso anterior *SubImg* y el filtro definido por *MFiltro*. Como valor de retorno se obtendrá en *r29* el valor final del píxel filtrado.

4. Devolverá el control al programa llamante, dando como resultado en *r29* el valor obtenido en el paso anterior.

Extracción de Submatriz

SubMatriz (Imagen, SubImg, i, j)

Parámetros:

- **Imagen:** Es la matriz que contiene la imagen de la que se ha de extraer una submatriz 3×3 . Se pasa por dirección, por lo que ocupa 4 bytes. Es un parámetro de entrada que tiene tres campos:
 - El número de filas, de tipo entero (M).
 - El número de columnas, de tipo entero (N).
 - Los elementos almacenados por filas, cada uno de tipo byte sin signo (MxN bytes, MxN/4 palabras).
- **SubImg:** Es la matriz de tamaño 3×3 en que la subrutina debe depositar la submatriz de la imagen de entrada que se determina tal como se especifica más adelante, en la descripción de esta subrutina. Se pasa por dirección, por lo que ocupa 4 bytes. Es un parámetro de salida que quedará relleno con los nueve valores (bytes sin signo) que forman la submatriz resultado.
- **i:** Es el número de fila (comprendido entre 0 y M-1) del píxel de la imagen proporcionada como primer parámetro *Imagen* que será el píxel central de la submatriz resultado. Es un parámetro de entrada que se pasa por valor.
- **j:** Es el número de columna (comprendido entre 0 y N-1) del píxel de la imagen proporcionada como primer parámetro *Imagen* que será el píxel central de la submatriz resultado. Es un parámetro de entrada que se pasa por valor.

Descripción:

La rutina **SubMatriz** obtiene una subimagen de tamaño 3×3 obtenida de la imagen de entrada *Imagen* sin más que copiar de la misma nueve valores de píxeles que se determinan tal como queda descrito en la explicación del funcionamiento de esta subrutina.

El funcionamiento de esta rutina será el siguiente:

1. Identificará si el píxel seleccionado, $I(i, j)$ (siendo *I* la imagen de entrada) pertenece al borde de la imagen. Siendo *M* el número de filas y *N* el número de columnas, el píxel $I(i, j)$ pertenecerá al borde en cualquiera de los siguientes casos: $i = 0$, $i = M - 1$, $j = 0$ o $j = N - 1$.
2. Si el píxel $I(i, j)$ pertenece al borde de la imagen, la submatriz resultado se rellenará con nueve valores idénticos al píxel $I(i, j)$ de la imagen de entrada.

3. Si el píxel $I(i, j)$ no pertenece al borde de la imagen, la submatriz resultado se rellenará con los siguientes nueve valores:

$$\begin{aligned} &I(i-1, j-1), I(i-1, j), I(i-1, j+1), \\ &I(i, j-1), I(i, j), I(i, j+1), \\ &I(i+1, j-1), I(i+1, j), I(i+1, j+1) \end{aligned}$$

Valor del píxel filtrado

`VPixel = ValorPixel( SubImg, MFiltro )`

Parámetros:

- **SubImg:** Es la matriz de tamaño 3×3 que contiene el conjunto de 9 valores (bytes sin signo) sobre los que se ha de aplicar el filtro especificado por el segundo parámetro. Es un parámetro de entrada que se pasa por dirección, por lo que ocupa 4 bytes.
- **MFiltro.** Es la matriz cuadrada que define el filtro que se ha de aplicar a la imagen. Es un parámetro de entrada que se pasa por dirección, por lo que ocupa 4 bytes. Define la posición de memoria en que se encuentra la especificación del filtro, formada por tres constantes enteras (con signo) cada una de las cuales ocupa una palabra:
 - La constante K , que divide a cada componente del filtro.
 - El valor C del elemento central del filtro.
 - El valor B de cada uno de los ocho elementos pertenecientes al borde de la matriz de filtro.

Valor de retorno:

- **VPixel:** La función devuelve el valor que se ha de asignar al píxel seleccionado una vez aplicado el filtro. El valor de retorno es un entero sin signo que se devuelve en el registro $r29$.

Descripción:

La rutina `ValorPixel` aplica el filtro *MFiltro* que se pasa como segundo parámetro a la submatriz que se pasa en el primer parámetro. Dada la estructura de ambos parámetros, la aplicación del filtro se limita a acumular sobre un registro los resultados parciales del producto de cada elemento de la submatriz por el correspondiente elemento del filtro, considerando en cada caso el formato de representación de dichos elementos (enteros con signo los del filtro, bytes sin signo los de la submatriz).

El funcionamiento de esta rutina será el siguiente:

1. Inicializará a 0 un acumulador entero de una palabra *Acc* y leerá los valores K , C y B (definidos como parte del filtro *MFiltro*), asignándoselos a distintas variables locales (pueden ser registros).
2. Asignará a un puntero la dirección del comienzo de la submatriz de entrada *SubImg*, es decir, la que apunta a su primer elemento.

3. Recorrerá en un bucle los 4 primeros elementos de la *SubImg*, multiplicando cada uno de ellos por la constante *B*, ya que estos cuatro elementos pertenecen al borde de la submatriz. En cada iteración de este bucle se efectuarán las siguientes operaciones:
 - a) Se leerá sobre un registro general el byte sin signo que determina el nivel de intensidad del píxel seleccionado en la submatriz. Con esta lectura, el registro contendrá el mismo valor pero representado como un entero de 32 bits.
 - b) Se multiplicará el valor obtenido en el paso anterior por el valor de *B*, acumulando el resultado sobre *Acc*.
4. Leerá el siguiente elemento de la submatriz (el valor central) sobre un registro general y lo multiplicará por el valor de *C*, acumulando el resultado sobre *Acc*.
5. Realizará las mismas operaciones del bucle del paso 3, pero ahora para los últimos cuatro elementos de *SubImg*, es decir, multiplicará cada uno de ellos por la constante *B*, ya que estos cuatro elementos también pertenecen al borde de la submatriz.
6. Dividirá el valor acumulado en *Acc* entre el valor de *K*, dejando de nuevo el resultado en *Acc*. Utilizará la división entera.
7. Ajustará el valor obtenido en el paso anterior, de modo que si es negativo se sustituirá por cero y si es positivo y superior a 255 se sustituirá por este nivel máximo de intensidad 255.
8. Devolverá el control al programa llamante, dando como resultado en *r29* el valor obtenido en el paso anterior.

Compara imágenes

Diferencia = Comp (Imagen1, Imagen2)

Parámetros:

- **Imagen1:** Es la matriz que contiene la primera de las imágenes que se han de comparar. Se pasa por dirección, por lo que ocupa 4 bytes. Es un parámetro de entrada que tiene tres campos:
 - El número de filas, de tipo entero (M).
 - El número de columnas, de tipo entero (N).
 - Los elementos almacenados por filas, cada uno de tipo byte sin signo (MxN bytes, MxN/4 palabras).
- **Imagen2:** Es la matriz que contiene la segunda de las imágenes que se han de comparar. Se pasa por dirección, por lo que ocupa 4 bytes. Es un parámetro de entrada que tiene tres campos con idéntica estructura que *Imagen1*.

Valor de retorno:

- **Diferencia:** La función devuelve una medida de la diferencia entre las dos imágenes que se han pasado como parámetro. Esta diferencia es la suma para todos los píxeles de los valores absolutos de la diferencia entre cada par de píxeles situados en la misma posición de *Imagen1* y *Imagen2*. El valor de retorno es un entero sin signo que se devuelve en el registro *r29*.

Descripción:

La rutina **Comp** recorre las dos imágenes a las que hacen referencia los parámetros y va acumulando el valor absoluto de la diferencia entre píxeles de una y otra imagen.

El funcionamiento de esta rutina será el siguiente:

1. Inicializará a 0 un acumulador de diferencias *Dif* (podrá alojarse en un registro).
2. Recorrerá los MxN elementos de cada matriz, sumando para cada uno de ellos al acumulador *Dif* el valor absoluto de la diferencia entre el píxel perteneciente a la primera imagen y el de la segunda.
3. Devolverá el control al programa llamante, dando como resultado en *r29* el valor del acumulador obtenido en el paso anterior.

Creación de una pila de usuario

Debido a que el 88110 no dispone de un registro de propósito específico para la gestión de la pila, se asignará como puntero de pila uno de los registros de propósito general. Se utilizará el registro **r30** como puntero de pila. Éste apuntará a la cima de la pila, es decir, a la palabra que ocupa la cabecera de la pila (donde estará la última información introducida) y ésta crecerá hacia direcciones de memoria decrecientes.

A modo de ejemplo se muestran las operaciones elementales a realizar sobre la pila: PUSH y POP (véase la Figura 3).

Supongamos que el registro **r30** contiene el valor 10000 (decimal) y el registro **r2** contiene el valor hexadecimal 0x04030201. La operación PUSH, que introduce este registro en la pila, se implementa con la siguiente secuencia de instrucciones:

```
subu r30,r30,4
st r2,r30,0
```

quedando $r30(SP) = 9996$ y la pila como se indica en la Figura 3.

Supongamos que después de realizar la operación anterior se realiza una operación POP sobre la pila con el registro **r3** como operando. La secuencia de instrucciones resultante sería la siguiente:

```
ld r3,r30,0
addu r30,r30,4
```

quedando $r3 = 0x04030201$ y $r30(SP) = 10000$

9996:	0x01
	0x02
	0x03
	0x04
10000:	x x x x
	x x x x
	x x x x
	x x x x

Figura 3. Operaciones PUSH y POP.

Subrutinas anidadas, variables locales y paso de parámetros

Puesto que las instrucciones de salto con retorno que proporciona el 88110 (**jsr** y **bsr**) salvaguardan la dirección de retorno en el registro **r1**, hay que incluir un mecanismo que permita realizar llamadas anidadas a subrutinas. Por este motivo es necesario guardar el contenido de dicho registro en la pila. Este mecanismo sólo es estrictamente necesario cuando una subrutina realiza una llamada a otra, pero, para sistematizar la realización de este proyecto, se propone realizar siempre a la entrada de una subrutina la salvaguarda del registro **r1** en la pila mediante una operación PUSH.

El espacio asignado para variables locales se reserva en el marco de pila de la correspondiente rutina. Para construir dicho marco de pila basta con asignar a uno de los registros de la máquina el valor del puntero de pila **r30**, después de haber salvaguardado el valor que tuviera el registro que actúa como puntero al marco de pila del llamante. Para la realización del proyecto se utilizará el registro **r31**. Por tanto, las primeras instrucciones de una subrutina que desea activar un nuevo marco de pila serán las siguientes:

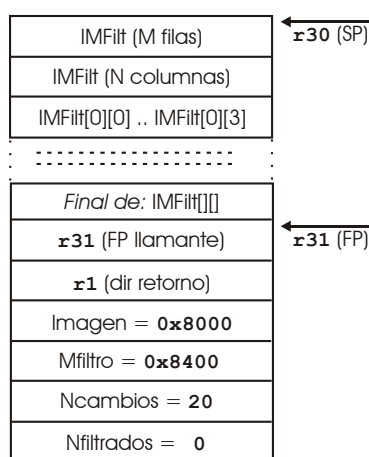
```

RUTINA:   subu r30,r30,4      ; Se realiza una operacion PUSH r1
          st r1,r30,0
          subu r30,r30,4      ; Se realiza una operacion PUSH r31
          st r31,r30,0
          addu r31,r30,r0     ; r31 <- r30

```

En este proyecto se utilizarán los dos métodos clásicos de paso de parámetros:

- **Paso por dirección:** Se pasa la dirección de memoria donde está contenido el valor del parámetro sobre el que la subrutina tiene que operar.
- **Paso por valor:** Se pasa el valor del parámetro sobre el que la subrutina tiene que operar.

Figura 4. Estado de la pila tras activarse la rutina `FiltRec`.

El paso de parámetros a todas las subrutinas del proyecto se realizará utilizando la pila de la máquina salvo cuando se especifique de otro modo. El parámetro que queda en la cima de la pila es el primero de la lista de argumentos. Por ejemplo, para una invocación de la rutina `FiltRec` los parámetros en pila quedarían tal y como se especifica en la Figura 4, en la que se ilustra también la reserva de espacio en la pila que realiza `FiltRec` para almacenar la imagen filtrada. El comienzo de esta rutina quedaría como sigue:

```
FiltRec:
    PUSH    (r1)           ; Se guarda la direccion de retorno
    PUSH    (r31)          ; Se salva el FP del llamante
    addu    r31,r30,r0      ; r31<-r30   Activación puntero de marco de pila
    ld      r10, r31, 8     ; lee los parámetros de la subrutina
    . . . . .              ; . . . . .
    clr     r6, r6, 2<0>    ; calcula y asigna a r6 el número de bytes a reservar
    subu    r30,r30,r6      ; r30<-r30-r6 Reserva espacio para la imagen filtrada
```

Asignación de etiquetas y de memoria

El punto de entrada de cada una de las subrutinas deberá ir asociado a las etiquetas `FiltRec`, `Filtro`, `FilPixel`, `SubMatriz`, `ValorPixel` y `Comp` (respetando el tipo de letra mayúsculas/minúsculas). Por ejemplo, la primera instrucción perteneciente a la subrutina `Comp` deberá ir precedida de esta etiqueta:

```
Comp: subu r30,r30,4
      .
      .
      .
```

El rango de direcciones **0x00007000 a la 0x00009000 se reservará para el programa corrector**, es decir, el alumno **no debe utilizar** dichas posiciones para almacenar código ni datos. Además, la pila se situará en las **posiciones altas de memoria** (por ejemplo, puede inicializarse el puntero de pila en la dirección 0x0000F000).

Por otra parte, cuando en una subrutina sea necesario utilizar memoria para almacenar temporalmente información, se empleará **siempre** memoria de la pila y **nunca** variables definidas en direcciones absolutas. Un proyecto que no se atenga a esta norma se evaluará como suspenso aún en el caso de que llegue a pasar las pruebas establecidas por el Departamento.

Ejemplos

A continuación se incluye una serie de ejemplos con los argumentos que se pasan a las distintas subrutinas y las direcciones de memoria que se modifican.

Todos los ejemplos se han descrito utilizando el formato de salida que ofrece el simulador del 88110. 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
```

Compara dos imágenes

Caso 1. Llamada a Comp

Llama a 'Comp', pasándole dos imágenes de 4x8 elementos que difieren en uno solo de ellos

r30=36856 (0x8FF8)

Direcciones de memoria:

```
36848      00800000      28800000

32768      04000000      08000000      00000000      00000000
32784      00000000      00070000      00000000      00000000
32800      00000000      00000000

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

Resultado:

r30=36856 (0x8FF8) r29=7 (0x7)

Caso 2. Llamada a Comp

Llama a 'Comp', pasándole dos imágenes de 4x8 elementos en las que difieren todos sus elementos en una unidad

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	FE54FE54	FE54FE54
32832	54FE54FE	54FE54FE	FE54FE54	FE54FE54

Resultado:

r30=36856 (0x8FF8) r29=32 (0x20)

Extracción de submatriz

Caso 3. 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.

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 4. 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 5. Llamada a ValorPixel

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

r30=36856 (0x8FF8)

Direcciones de memoria:

36848			00800000	10800000
32768	00000000	55000000	00000000	
32784	01000000	01000000	00000000	

Resultado:

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

Caso 6. Llamada a ValorPixel

Llama a 'ValorPixel', pasándole una subimagen no nula y un filtro que devuelve la media de los ocho elementos que le rodean.

r30=36856 (0x8FF8)

Direcciones de memoria:

36848			00800000	10800000
32768	10111213	14151617	18000000	
32784	08000000	00000000	01000000	

Resultado:

r30=36856 (0x8FF8) r29=20 (0x14)

Filtro de un píxel

Caso 7. Llamada a FilPixel

Llama a 'FilPixel', pasándole una imagen de 4x8 elementos y un filtro unitario que se aplica al único elemento no nulo.

r30=36848 (0x8FF0)

Direcciones de memoria:

36848	00800000	01000000	01000000	30800000
32768	04000000	08000000	00000000	00000000
32784	00330000	00000000	00000000	00000000
32800	00000000	00000000		
32816	01000000	01000000	00000000	

Resultado:

r30=36848 (0x8FF0) r29=51 (0x33)

Caso 8. 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	44353433	44444444	44448844	44444444
32800	44444444	44444444		
32816	08000000	00000000	01000000	

Resultado:

r30=36848 (0x8FF0) r29=62 (0x3E)

Filtro de imagen

Caso 9. Llamada a Filtro

Llama a 'Filtro', pasándole una imagen no nula de 4x8 elementos y un filtro que devuelve la imagen duplicando el valor de los elementos.

r30=36852 (0x8FF4)

Direcciones de memoria:

36848		00800000	28800000	50800000
32768	04000000	08000000	78563412	44332211

32784	78563412	45342312	78563412	44332211
32800	78563412	45342312		
32800			00000000	00000000
32816	00000000	00000000	00000000	00000000
32832	00000000	00000000	00000000	00000000
32848	01000000	02000000	00000000	

Resultado:

r30=36852 (0x8FF4)

Direcciones de memoria:

32800			04000000	08000000
32816	F0AC6824	88664422	F0AC6824	8A684624
32832	F0AC6824	88664422	F0AC6824	8A684624

Caso 10. Llamada a Filtro

Llama a 'Filtro', pasándole una imagen no nula de 4x8 elementos y un filtro que multiplica por 4 cada elemento. Algunos alcanzan el valor máximo.

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	01000000	04000000	00000000	

Resultado:

r30=36852 (0x8FF4)

Direcciones de memoria:

32800			04000000	08000000
32816	04080C10	1014181C	44FF4C50	50FF585C
32832	84888C90	FF94989C	C4C8CCD0	D0D4D8DC

Filtro recursivo de una imagen

Caso 11. Llamada a FiltRec

Llama a 'FiltRec', pasándole una imagen no nula de 4x8 elementos y un filtro que sustituye cada elemento por la media de los que le rodean. El parámetro NCambios tiene valor 2.

r30=36848 (0x8FF0)

Direcciones de memoria:

36848	00800000	30800000	02000000	00000000
32768	04000000	08000000	00FF00FF	FF00FF00
32784	00FF00FF	FF00FF00	00FF00FF	FF00FF00
32800	00FF00FF	FF00FF00		
32816	08000000	00000000	01000000	

Resultado:

r30=36848 (0x8FF0) r29=9 (0x9)

Direcciones de memoria

32768	04000000	08000000	00FF00FF	FF00FF00
32784	004C8B9D	9D8B4C00	004C8B9D	9D8B4C00
32800	00FF00FF	FF00FF00		

Caso 12. Llamada a FiltRec

Llama a 'FiltRec', pasándole una imagen no nula de 4x8 elementos y un filtro que sustituye cada elemento por la media de los que le rodean. El parámetro NCambios tiene valor 0, por lo que se detiene al alcanzar el máximo número de llamadas recursivas (15 filtrados).

r30=36848 (0x8FF0)

Direcciones de memoria:

36848	00800000	30800000	00000000	00000000
32768	04000000	08000000	FF0000FF	FF0000FF
32784	FF0000FF	FF0000FF	FF0000FF	FF0000FF
32800	FF0000FF	FF0000FF		
32816	08000000	00000000	01000000	

Resultado:

r30=36848 (0x8FF0) r29=15 (0x0F)

Direcciones de memoria

32768	04000000	08000000	FF0000FF	FF0000FF
32784	FF8F7795	95778FFF	FF8F7795	95778FFF
32800	FF0000FF	FF0000FF		

NORMAS DE PRESENTACIÓN

Toda la información relativa a este proyecto se encuentra disponible en:

http://www.datsi.fi.upm.es/docencia/Estructura_09/Proyecto_Ensamblador

Esta página contiene una sección de anuncios relacionados con el proyecto.

CONVOCATORIA DE JUNIO 2015

El plazo de entrega del proyecto estará abierto desde el viernes día **27 de marzo** hasta el lunes **4 de mayo de 2015** en que se realizará la corrección definitiva. A partir del día 6 de mayo el sistema estará configurado de modo que permita entregar únicamente la memoria del proyecto. El plazo de entrega para dicha memoria finalizará el viernes 8 de mayo a las 21:00.

Cada grupo podrá disponer de las correcciones que se realizarán los días **27 de marzo, 7 de abril y 4 de mayo**, así como un máximo de **cinco** de las correcciones del proyecto planificadas para los días 8 a 10, 13 a 17, 20 a 24 y 27 a 30 de abril.

Todas las correcciones se realizarán a partir de las 21:00. Para solicitar una corrección bastará con entregar correctamente los ficheros del proyecto antes de dicha hora límite.

El examen del proyecto está planificado para el viernes día 8 de mayo a las 17:00 (dentro del horario habilitado por Jefatura de Estudios para evaluación).

CONVOCATORIA EXTRAORDINARIA DE JULIO 2015

Todos los alumnos que se presenten a esta convocatoria deberán realizar la **entrega completa de los ficheros del proyecto**, independientemente de que se hayan podido presentar en alguna de las convocatorias anteriores y del número de pruebas que previamente hubieran superado.

El plazo de entrega del proyecto para esta convocatoria terminará el viernes día **26 de junio de 2015 a las 21:00**, en que se realizará una corrección para todos los grupos que hayan realizado la entrega y se encuentren pendientes de corrección. Además, el viernes 19 y los días 22 a 25 de junio se realizarán correcciones a las 21:00 para todos los alumnos que hayan realizado correctamente la entrega del proyecto. A partir del día 1 de julio el sistema estará configurado de modo que permita entregar únicamente la memoria del proyecto. El plazo de entrega para dicha memoria finalizará el lunes 6 de julio a las 20:00.

El examen del proyecto se realizará el mismo día del examen extraordinario de teoría (lunes 29 de junio) y a continuación de este. La hora de comienzo se anunciará en la Web de la asignatura pero, dado que el examen de teoría está planificado para las 15:00, es previsible que el del proyecto no comience antes de las 17:30.

TODAS LAS CONVOCATORIAS

La última entrega del proyecto no tiene carácter “obligatorio”. Es decir, una vez que los ficheros entregados por un grupo superan todas las pruebas del proyecto, no es necesario que el grupo realice una nueva entrega para la corrección del último día. En caso de no haber entregado la versión definitiva del fichero que contiene la memoria del proyecto, se podrá entregar en los días siguientes al de la última corrección (consulte las fechas concretas en el apartado de cada convocatoria o en la Web del proyecto).

Salvo que se indique lo contrario en la Web de la asignatura, durante el examen del proyecto se permitirá utilizar una copia en papel de los **documentos originales** (sin

ninguna anotación) que facilita el Departamento: enunciado del proyecto y manual del 88110. Por ello se recomienda que cada alumno disponga de su propia copia de estos documentos.

Durante la realización del examen o en los días previos se indicará si se permite el uso de calculadora, que en cualquier caso se referiría a un modelo básico, no programable. En ningún caso se permitirá el uso de teléfonos móviles ni otros dispositivos con capacidad de almacenamiento y/o conexión remota.

TUTORÍAS DEL PROYECTO

Las preguntas relacionadas con este proyecto se atenderán por correo electrónico en la dirección (pr_ensamblador@datsi.fi.upm.es) y personalmente en los despachos 4105 y/o 4106. El horario de atención personal a los alumnos para cuestiones relacionadas con este proyecto es el indicado en la web del Departamento para los profesores encargados del proyecto durante el presente curso académico (José L. Pedraza, Manuel M. Nieto):

<http://www.datsi.fi.upm.es/docencia/tutorias.html>

ENTREGA DEL PROYECTO

La entrega se compone de:

1. Una **memoria**, en formato DIN-A4, en cuya portada deberá figurar claramente el nombre y apellidos de los **autores** del proyecto, identificador del **grupo de alumnos** (el mismo que emplean para realizar las entregas y consultas) y el nombre de la asignatura.

Dicha memoria se entregará en formato electrónico, como un fichero PDF, mediante el sistema de entregas. Contendrá al menos los siguientes apartados:

- Histórico del desarrollo de las rutinas, con fechas, avances y dificultades encontradas, especificando el trabajo que realiza cada miembro del grupo o si dicho trabajo es común. Se detallará en este apartado el número total de horas invertidas en el proyecto por cada miembro del grupo, así como la relación de visitas realizadas a los profesores del proyecto.
- Descripción resumida del juego de ensayo (conjunto de casos de prueba) que el grupo haya diseñado y utilizado para probar el correcto funcionamiento del proyecto.
- Observaciones finales y comentarios personales de este proyecto, entre los que se debe incluir una descripción de las principales dificultades surgidas para su realización.

NOTA: Esta memoria no debe incluir el listado en ensamblador del código generado por el grupo, por lo que sí será necesario que todas las subrutinas se encuentren adecuadamente comentadas en el propio fichero que se entrega para su corrección automática (el descrito en el siguiente punto, *filtror.ens*), ya que dicho fichero será consultado en el proceso de evaluación del proyecto.

2. La entrega de los ficheros que contienen el proyecto. Será obligatorio entregar los siguientes ficheros:

- **autores:** Es un fichero ASCII que deberá contener los apellidos, nombre, número de matrícula, DNI y dirección de correo electrónico de los autores del proyecto. El proyecto se realizará individualmente o en grupos de **dos alumnos**. Cada línea de este fichero contendrá los datos de uno de los autores de acuerdo al siguiente formato:

Nº Matrícula; DNI ; apellido apellido, nombre; correo_electrónico

El número de matrícula que se debe indicar en el fichero es el que **asigna la secretaría del Centro** (por ejemplo 990999) y no el que se utiliza como identificador para abrir cuentas en el Centro de Cálculo (por ejemplo a990999). La dirección de correo electrónico deberá ser una dirección válida que el alumno consulte frecuentemente, ya sea la asignada por la UPM o bien otra dirección personal.

- **filtror.ens:** Contendrá las subrutinas que componen el proyecto junto con un programa principal que se haya utilizado para su depuración.
- **memoria.pdf:** Será un fichero en formato PDF que contenga la memoria del proyecto. En las entregas previas a la definitiva, el contenido del fichero **memoria.pdf** podrá limitarse a la identificación de los componentes del grupo que realiza el proyecto.

IMPORTANTE: Antes de efectuar cada entrega del proyecto se recomienda realizar el ensamblado del fichero **filtror.ens** asegurando que no genera ningún error, así como ejecutar el código del proyecto con varios casos de prueba. De este modo minimizará la probabilidad de malgastar alguna de las correcciones disponibles.

FORMA DE ENTREGA DE LOS FICHEROS

Los ficheros del proyecto se pueden entregar desde **batman.fi.upm.es** mediante un programa de entrega específico ("**ent_88k**"), desde la red de PCs del Centro de Cálculo mediante un programa disponible a través de **g:\datsi\entregas\ent_88k** o bien mediante el entregador Web disponible en la dirección:

<http://www.datsi.fi.upm.es/Practicas>

Se recomienda el uso de esta última opción, que puede utilizar mediante cualquiera de los navegadores convencionales.

En cualquiera de los casos, e independientemente del método de entrega que utilice, recuerde que la información de su proyecto no debe compartirla con otros alumnos o grupos, así como que debe evitar que puedan obtenerla por cualquier método, ya que en caso contrario su trabajo podría verse involucrado en un caso de COPIA, cuyas consecuencias están descritas en las normas de la asignatura. Por ello le recomendamos que siempre que trabaje en equipos de la Escuela o, en general, en cualquier equipo al que puedan acceder directa o indirectamente otros alumnos, lo haga siempre manteniendo los ficheros del proyecto en dispositivos privados, (extraíbles), evitando así dejar copias temporales en lugares accesibles por otros grupos.

Apéndice A

Instalación del paquete de prácticas y proyectos 88110 en un computador personal

Si dispone de un computador personal con Windows, Linux o Solaris, puede descargar y utilizar una versión del emulador y del ensamblador que se pueden ejecutar bajo estos sistemas operativos. Además, si dispone de conexión a Internet, podrá realizar la entrega de la práctica o proyecto desde dicho computador.

A.0.1. Obtención del entorno de prácticas y/o proyectos

La distribución del entorno para la realización de prácticas y proyectos se puede obtener a través del URL

http://www.datsi.fi.upm.es/docencia/Estructura_09/Proyecto_Ensamblador.

En esta página hay distintas versiones en función del sistema operativo que se tenga instalado, ya sea Windows, Solaris o Linux.

A.1. Instalación del paquete bajo Linux o Solaris

Los pasos que se deben seguir para realizar la instalación del entorno de prácticas y proyectos son iguales para los sistemas operativos Linux o Solaris, por lo que se describirá a continuación la instalación bajo Linux:

- Entre en el computador Linux como administrador (root).
- Copie el fichero `88k_Linux_Static.tar.gz` que ha obtenido en un directorio del disco duro del sistema Linux (p.e. `/tmp`).
- Sitúese en el directorio donde ha copiado el fichero `88k_Linux_Static.tar.gz` y descomprímalo:

```
cd /tmp
tar zxvf 88k_Linux_Static.tar.gz
```

Este comando habrá generado el directorio EM88110, que contendrá todos los ficheros de la distribución.

- Cambie el directorio de trabajo, situándose en el directorio que acaba de crear:

```
cd EM88110
```

- Si teclea el comando `ls -l` obtendrá un listado con todos los ficheros que componen la distribución. Entre ellos se encuentra el fichero `INSTALL`. Para instalar el paquete ejecute:

```
sh INSTALL
```

- El comando de instalación le preguntará por dos directorios donde ubicar la instalación:

```
Introduzca el directorio donde se va a instalar el emulador
(por defecto /usr/local/88k)
```

En este directorio se copiarán los ficheros que componen la herramienta. Debe introducir un camino completo dentro del árbol de directorios, o simplemente pulsar la tecla *enter* para aceptar el directorio por defecto. A continuación aparecerá el siguiente mensaje:

```
El directorio donde se van a instalar los ejecutables debe formar
parte de la variable PATH. Introduzca el directorio donde se van a
instalar los ejecutables (por defecto /usr/bin)
```

En este caso debe especificar el directorio donde se van a buscar los ejecutables de la instalación. Este directorio debe estar referenciado en la variable de entorno `PATH`. Al igual que en el caso anterior si pulsa *enter* se acepta la opción por defecto. A continuación aparecerá el siguiente mensaje:

```
El directorio de instalacion es /usr/local/88k
El directorio de los ejecutables es /usr/bin
Si los parametros son correctos pulse enter
```

Si pulsa *enter* se continúa con la instalación y si los dos directorios que ha especificado son correctos el programa se habrá instalado correctamente. Si, por el contrario, desea cambiar alguno de los parámetros, pulse `CTRL C` y se cancelará la instalación.

- Si la instalación se ha efectuado correctamente, salga de la cuenta `root`, vuelva a entrar como un usuario no privilegiado y ejecute los comandos de la práctica o el proyecto. Deben funcionar correctamente.

A.2. Instalación del paquete bajo Windows

Se describe a continuación el procedimiento que se debe seguir para realizar la instalación del entorno de prácticas y proyectos bajo Windows. La versión del sistema operativo Windows es indiferente ya que en realidad se trata de una aplicación MS-DOS que se ejecuta en una ventana.

- Copie el fichero `88k.Windows.zip` en una carpeta cualquiera del sistema Windows, por ejemplo en el Escritorio.
- Descomprima la carpeta `88k.Windows.zip` sobre una ubicación que le resulte cómoda para trabajar desde una ventana MS-DOS, por ejemplo en:

C : \GradoII\Ensamblador

- Abra una ventana MS-DOS. Lo puede hacer, por ejemplo, pulsando “Inicio” seguido de “Ejecutar” y especificar el comando “cmd”. Al pulsar “Intro” se abrirá una ventana.
- Sitúese en la carpeta en que ha descomprimido `88k.Windows.zip`, haciendo:

cd C : \GradoII\Ensamblador\88k.Windows

si ha utilizado la ubicación propuesta anteriormente.

- Trabaje directamente en esta ventana, ejecutando los programas `88110e` y `mc88110` tal como se ha descrito previamente. Tenga en cuenta que si bien el ensamblado y la ejecución se deben realizar desde la ventana MS-DOS, la edición del código puede efectuarla con cualquier editor que no introduzca caracteres de control en el fichero fuente. Por ejemplo, puede utilizar “Wordpad” o mejor aún el editor de libre distribución “Notepad++”, pero no debe emplear procesadores de texto como “OpenOffice Writer” o “Microsoft Word”.