Proyecto de Programación en Ensamblador
Estructura de Computadores, Plan 2009

Departamento de Arquitectura y Tecnología de Sistemas Informáticos

Primer Semestre, 2013-2014

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 constante K y una matriz de dimensión 3. Cada uno de los elementos del filtro es en realidad un número fraccionario que se obtiene al dividir el correspondiente elemento de la matriz entre la constante K.

AVISO –2013/2014–

El enunciado de este proyecto coincide con el planteado el semestre/curso anterior, por lo que aquellos alumnos que tengan que repetir o corregir el proyecto que desarrollaron entonces, podrán partir de los programas ya realizados anteriormente, si bien *deberán superar las pruebas que se establezcan para la convocatoria actual*.

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 el mismo proyecto que en el 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, ya que es la opción que realiza por defecto el procesador simulado. Además, puesto que los píxeles de las dos filas y de las dos columnas de los extremos de la imagen no tienen ocho vecinos, se evita aplicar para dichos píxeles las operaciones de filtrado.

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 (una vez que se ha dividido por la constante K), $m_{i,j}$ 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 $r_{2,2} = 4$:

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

Para obtener esta expresión en que cada coeficiente del filtro tiene el valor $0,125$, se ha partido de los coeficientes de la matriz Mat, todos los cuales tienen valor unidad y se han dividido entre la constante $K = 8$.

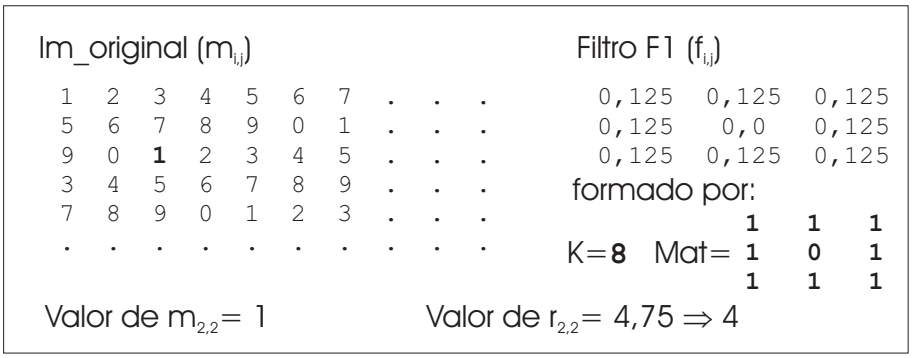

Figura 1. Aplicación de un filtro de imagen.

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 se ha modificado un número suficientemente alto de píxeles (se definirá más adelante).
- El número de veces que se ha aplicado el filtro no ha alcanzado 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 cuatro subrutinas que se relacionan tal y como se indica en la figura 2. Además, se construirá un programa principal para cada una de las pruebas que sea necesario realizar 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.

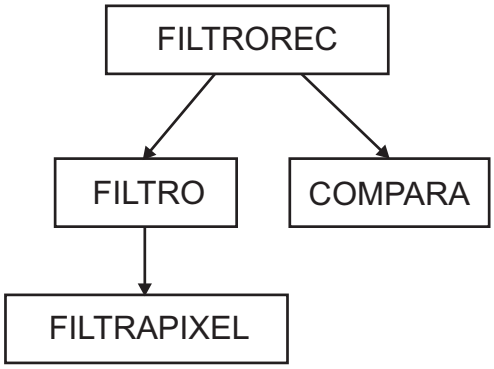

Figura 2. Jerarquía de las rutinas del proyecto.

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.

Filtro recursivo

```
NPasadas = FILTROREC ( Imagen, MFiltro, NCambios, NPasadas )
```

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. El filtro genérico que se ha de aplicar de forma recursiva está especificado por la matriz de coeficientes que se facilita en el segundo parámetro. 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 los elementos del filtro que se ha de pasar a la imagen proporcionada a través del primer parámetro. 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 dos campos:
  - La constante entera K que divide a cada componente del filtro
  - Los nueve elementos del filtro almacenados por filas, cada uno de tipo entero (en total 9 palabras).
- **NCambios:** Es un valor que define el mínimo número de píxeles diferentes 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.
- **NPasadas:** Es un valor que define el número de pasadas ya realizadas del filtro definido por $MFiltro$. Es un parámetro de entrada, de tipo entero, se pasa por valor y ocupa 4 bytes.

Valor de retorno:

- **NPasadas:** La función devuelve el número de pasadas del filtro ya realizadas a la imagen original. El valor de retorno se devuelve en el registro $r29$.

Descripción:

La rutina `FILTROREC` 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:

1. El número de pasadas ya realizadas sobrepasa el valor límite, que se ha fijado en **20**. Puesto que este número se comunica como parámetro de entrada, es necesario que el programa llamante a esta subrutina proporcione `0` como valor de este parámetro.

2. El número de diferencias detectadas entre la imagen original y la filtrada tras la última aplicación del filtro es menor que el fijado por 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`, que se encargará de aplicar una vez el filtro definido por la matriz de coeficientes $MFiltro$. Para ello habrá preparado los parámetros de dicha llamada, que serán los siguientes: la imagen original $Imagen$, la zona de memoria de la imagen filtrada, que será la que se ha reservado en el marco de pila en el paso anterior $ImFiltrada$, y la matriz de coeficientes, $MFiltro$.

3. Incrementará el número de pasadas del filtro que se ha realizado, que pasará a ser $NPasadas + 1$. Si $(NPasadas + 1) >= 20$, se abandonará el proceso recursivo (por requerir un número demasiado alto de aplicaciones del filtro). En este caso, la ejecución continuará en el paso 6 de este procedimiento.

4. Contabilizará el número de píxeles que difieren entre la imagen original y la filtrada para decidir si el proceso recursivo debe continuar o finalizar. Para ello, llamará a la subrutina `COMPARA`, pasándole como parámetros las imágenes original y filtrada. Si el número de cambios detectado por `COMPARA` es menor que el especificado por el parámetro $NCambios$, se dará por terminado el proceso recursivo, continuando la ejecución en el paso 6 de este procedimiento.

5. Al no haberse alcanzado el final de la recursividad, llamará a la propia subrutina `FILTROREC`, con los siguientes parámetros: la imagen filtrada, $ImFiltrada$, la matriz de coeficientes, $MFiltro$, el número máximo de cambios para que se termine el proceso recursivo $NCambios$ y, finalmente, el número de pasadas ya realizadas, que se encontró en el paso 3 de este procedimiento. Al retornar de esta llamada recursiva, la ejecución continuará por el paso 6.

6. Copiará la imagen filtrada sobre la zona que ocupaba la imagen original.

7. Devolverá el control al programa llamante, proporcionándole como valor de retorno el número de pasadas realizadas hasta el momento.

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. El filtro genérico que se ha de aplicar a la imagen está especificado por la matriz de coeficientes que se facilita en el último parámetro. 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 la imagen de entrada una vez que se le haya aplicado el filtro. 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 los elementos del filtro que se ha de pasar a la imagen proporcionada a través del primer parámetro. 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 dos campos:
  - La constante entera K que divide a cada componente del filtro
  - Los nueve elementos del filtro almacenados por filas, cada uno de tipo entero (en total 9 palabras).

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. Copiará la primera fila completa de la imagen original sobre la imagen filtrada.
3. Desde la segunda ($i = 1$) hasta la penúltima fila ($i = (M-2)$), de la matriz original, realizará las siguientes operaciones:
   a) Copiará el píxel de la primera columna ($j = 0$) desde la imagen original hasta la imagen filtrada.
   b) Iniciará un bucle que recorrerá desde el segundo ($j = 1$) al penúltimo ($j = (N-2)$) píxel de la fila que se está procesando en que se realizarán las siguientes operaciones:

1) Preparará los parámetros y realizará una llamada a la subrutina `FILTRAPIXEL` 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 matriz de coeficientes $MFiltro$.
2) Almacenará el valor de retorno recogido de la subrutina `FILTRAPIXEL` en la posición que le corresponde (i, j) de la imagen filtrada $ImFiltrada$.

c) Copiará el píxel de la última columna de la fila actual, desde la imagen original hasta la imagen filtrada.

4. Copiará la última fila completa de la imagen original sobre la imagen filtrada.

Filtro de un píxel

```
ValorPixel = FILTRAPIXEL ( 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. El filtro genérico que se ha de aplicar a la imagen está especificado por la matriz de coeficientes que se facilita en el último parámetro. 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).
- **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 los elementos del filtro que se ha de pasar a la imagen proporcionada a través del primer parámetro. 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 dos campos:
  - La constante entera K que divide a cada componente del filtro
  - Los nueve elementos del filtro almacenados por filas, cada uno de tipo entero (en total 9 palabras).

Valor de retorno:

- **ValorPixel:** 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 `FILTRAPIXEL` 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 dan lugar a un elemento válido de la imagen, sino que se supondrá que el programa llamante solo proporcionará valores adecuados para dichos parámetros.

El funcionamiento de esta rutina será el siguiente:

1. Inicializará a 0 un acumulador entero de una palabra ACC y leerá el valor K (definido como parte del filtro $MFiltro$), asignándoselo a una variable local (puede ser un registro).

2. Asignará a un puntero la dirección del elemento $i-1, j-1$ de la imagen de entrada $Imagen$. Otro puntero señalará al primer elemento de la matriz de coeficientes del filtro $MFiltro$.

3. Recorrerá los 9 elementos de la $Imagen$ involucrados en la operación de filtro tal como se explica en la introducción de este enunciado. Al mismo tiempo se recorren los 9 elementos de la matriz de coeficientes del filtro. En cada iteración de estos bucles 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. 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 correspondiente de la matriz de coeficientes, acumulando el resultado sobre el acumulador ACC.

4. Dividirá el valor acumulado en ACC entre el valor de K, dejando de nuevo el resultado en ACC. Utilizará la división entera.

5. 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.

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

Compara imágenes

```
Diferencias = COMPARA ( Imagen1, Imagen2 )
```

Parámetros:

- **Imagen1:** Es la matriz que contiene la primera de las dos 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 dos 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:

- **Diferencias:** La función devuelve el número de píxeles cuyo valor de intensidad difiere entre las dos imágenes proporcionadas. El valor de retorno es un entero sin signo que se devuelve en el registro $r29$.

Descripción:

La rutina `COMPARA` recorre las dos imágenes a las que hacen referencia los parámetros y cuenta el número de píxeles que difieren entre ambas.

El funcionamiento de esta rutina será el siguiente:

1. Inicializará a 0 un contador de diferencias Dif.
2. Recorrerá los MxN elementos de cada matriz, incrementando el contador cada vez que el píxel correspondiente a ambas sea distinto.
3. Devolverá el control al programa llamante, dando como resultado en $r29$ el valor 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
```

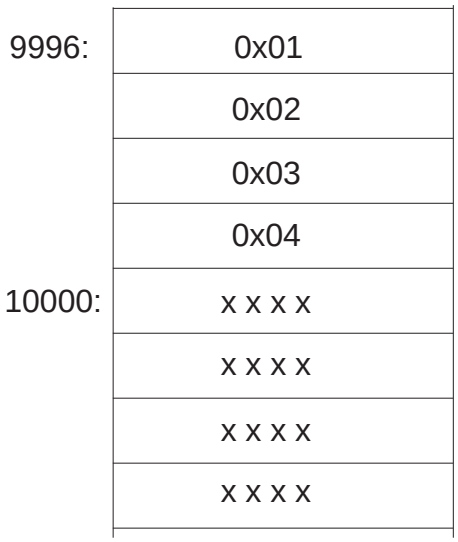

Figura 3. Operaciones PUSH y POP.

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

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
```

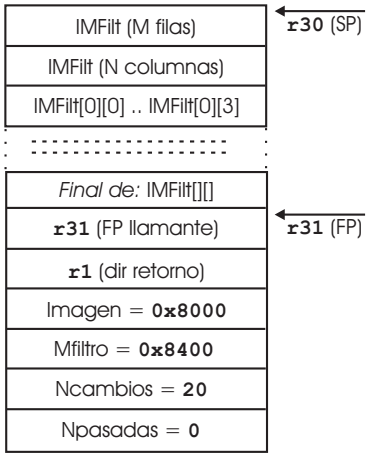

Figura 4. Estado de la pila tras activarse la rutina `FILTROREC`.

```
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.

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 `FILTROREC` 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 `FILTROREC` para almacenar la imagen filtrada. El comienzo de esta rutina quedaría como sigue:

```
FILTROREC:
    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 `FILTROREC`, `FILTRO`, `FILTRAPIXEL` y `COMPARA` (todas las letras en mayúsculas). Por ejemplo, la primera instrucción perteneciente a la subrutina `COMPARA` deberá ir precedida de esta etiqueta:

```
COMPARA: subu r30,r30,4
              .
              .
              .
```

El rango de direcciones **0x00007000 a la 0x00009000 se reservará para el programa corrector**, es decir, el alumno **no debe ubicar** en dichas posiciones ni código ni datos. 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
```

Filtrado de un píxel

Caso 1. Llamada a `FILTRAPIXEL`

Se realiza el filtrado de un píxel en una imagen nula y con un filtro unitario, que deja cada píxel con el mismo valor que tenía antes del filtrado.

```
r30=40944 (0x9FF0)
   Direcciones de memoria:
   40944      00800000     02000000     02000000     28800000

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

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

Resultado:

```
r30=40944 (0x9FF0)  r29=0 (0x0)
```

Caso 2. Llamada a `FILTRAPIXEL`

Se realiza el filtrado de un píxel en una imagen no nula con un filtro que duplica el valor del elemento sobre el que se aplica.

```
r30=40944 (0x9FF0)
   Direcciones de memoria:
   40944      00800000     02000000     02000000     28800000

   32768      04000000     08000000     44444444     44444444
   32784      44444444     44444444     44444444     44444444
   32800      44444444     4444444

   32800                                01000000     00000000
   32816      00000000     00000000     00000000     02000000
   32832      00000000     00000000     00000000     00000000
```

Resultado:

```
r30=40944 (0x9FF0)  r29=136 (0x88)
```

Caso 3. Llamada a `FILTRAPIXEL`

Se realiza el filtrado de un píxel en una imagen no nula con un filtro que obtiene la media de los tres elementos situados en la fila anterior.

```
r30=40944 (0x9FF0)
   Direcciones de memoria:
   40944      00800000     02000000     02000000     28800000

   32768      04000000     08000000     44444444     44444444
   32784      44353433     44444444     44448844     44444444
   32800      44555453     44444444

   32800                                03000000     01000000
   32816      01000000     01000000     00000000     00000000
   32832      00000000     00000000     00000000     00000000
```

Resultado:

```
r30=40944 (0x9FF0)  r29=52 (0x34)
```

Filtrado de una imagen

Caso 4. Llamada a `FILTRO`

Llama a FILTRO pasándole una imagen nula de 4x8 elementos y un filtro unitario que devuelve la misma imagen recibida.

```
r30=40948 (0x9FF4)
   Direcciones de memoria:
   40944                  00800000    28800000    50800000

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

   32848     01000000     00000000    00000000    00000000
   32864     00000000     01000000    00000000    00000000
   32880     00000000     00000000
```

Resultado:

```
r30=40948 (0x9FF4)
   Direcciones de memoria:
   32800                              04000000    08000000
   32816     00000000     00000000    00000000    00000000
   32832     00000000     00000000    00000000    00000000
```

Caso 5. Llamada a `FILTRO`

Llama a FILTRO pasándole una imagen no nula de 4x8 elementos y un filtro que devuelve la imagen dividiendo entre 2 el valor de sus elementos.

```
r30=40948 (0x9FF4)
   Direcciones de memoria:
   40944                  00800000    28800000    50800000

   32768     04000000     08000000    78563412    FBFCFDFE
   32784     78563412     FBFCFDFE    78563412    FBFCFDFE
   32800     78563412     FBFCFDFE

   32848     02000000     00000000    00000000    00000000
   32864     00000000     01000000    00000000    00000000
   32880     00000000     00000000
```

Resultado:

```
r30=40948 (0x9FF4)
   Direcciones de memoria:
   32800                              04000000    08000000
   32816     78563412     FBFCFDFE    782B1A09    7D7E7EFE
   32832     782B1A09     7D7E7EFE    78563412    FBFCFDFE
```

Compara dos imágenes

Caso 6. Llamada a `COMPARA`

Llama a COMPARA, pasándole dos imagenes de 4x8 elementos que difieren en uno solo de ellos

```
r30=40952 (0x9FF8)
   Direcciones de memoria:
   40944                                 00800000     28800000

   32768      04000000     08000000      00000000     00000000
   32784      00000000     00010000      00000000     00000000
   32800      00000000     00000000

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

Resultado:

```
r30=40952 (0x9FF8)  r29=1 (0x1)
```

Caso 7. Llamada a `COMPARA`

Llama a COMPARA, pasándole dos imagenes de 4x8 elementos en las que difieren todos sus elementos

```
r30=40952 (0x9FF8)
   Direcciones de memoria:
   40944                                 00800000     28800000

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

   32800                                 04000000     08000000
   32816      FF77FF77     FF77FF77      77FF77FF     77FF77FF
   32832      FF77FF77     FF77FF77      77FF77FF     77FF77FF
```

Resultado:

```
r30=40952 (0x9FF8)  r29=32 (0x20)
```

Filtrado recursivo de una imagen

Caso 8. Llamada a `FILTROREC`

Llama a FILTROREC pasándole una imagen no nula de 4x8 elementos y un filtro que sustituye cada elemento por la media de los que le rodean. El número de cambios admitido es 2.

```
r30=40944 (0x9FF0)
   Direcciones de memoria:
   40944      00800000     28800000      02000000     00000000

   32768      04000000     08000000      00FF00FF     FF00FF00
```

```
   32784      00FF00FF     FF00FF00     00FF00FF     FF00FF00
   32800      00FF00FF     FF00FF00

   32800                                08000000     01000000
   32816      01000000     01000000     01000000     00000000
   32832      01000000     01000000     01000000     01000000
```

Resultado:

```
r30=40944 (0x9FF0)  r29=9 (0x9)

   Direcciones de memoria
   32768      04000000     08000000     00FF00FF     FF00FF00
   32784      004C8B9D     9D8B4C00     004C8B9D     9D8B4C00
   32800      00FF00FF     FF00FF00
```

Caso 9. Llamada a `FILTROREC`

Llama a FILTROREC pasándole una imagen no nula de 4x8 elementos y un filtro que sustituye cada elemento por la media de los que le rodean. El número de cambios admitido es 0, por lo que se detiene al alcanzar el máximo número de iteraciones (20).

```
r30=40944 (0x9FF0)
   Direcciones de memoria:
   40944      00800000     28800000     00000000     00000000

   32768      04000000     08000000     FF0000FF     FF0000FF
   32784      FF0000FF     FF0000FF     FF0000FF     FF0000FF
   32800      FF0000FF     FF0000FF

   32800                                08000000     01000000
   32816      01000000     01000000     01000000     00000000
   32832      01000000     01000000     01000000     01000000
```

Resultado:

```
r30=40944 (0x9FF0)  r29=20 (0x14)

   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 FEBRERO 2014

El plazo de entrega del proyecto estará abierto desde el lunes día **28 de octubre** hasta el lunes **25 de noviembre de 2013** en que se realizará la corrección definitiva. A partir del día 28 de noviembre el sistema estará configurado de modo que permita entregar únicamente la memoria del proyecto. El plazo de entrega para dicha memoria finalizará el miércoles 4 de diciembre a las 20:00.

Cada grupo podrá disponer de las correcciones que se realizarán los días **28 de octubre, 4 y 25 de noviembre**, así como un máximo de **cinco** de las correcciones del proyecto planificadas para los días 5 a 8, 11 a 15 y 18 a 22 de noviembre.

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 miércoles día 4 de diciembre en el horario habilitado por Jefatura de Estudios para evaluación (14:00).

CONVOCATORIA EXTRAORDINARIA DE JULIO 2014

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

El plazo de entrega del proyecto para esta convocatoria terminará el viernes día **27 de junio de 2014 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 20 y los días 23 a 26 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 7 de julio a las 20:00.

El examen del proyecto se realizará el mismo día del examen extraordinario de teoría (martes 1 de julio) 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, *filtro.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 de la Facultad** (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. Se recomienda utilizar la dirección oficial asignada al alumno por la UPM o por la FI, aunque se admiten otras direcciones personales.
   - **filtro.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 `filtro.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

Sistema de entrega en batman

Se utilizará un programa de entrega denominado **"ent_88k"**. Para ejecutar este programa se deberá teclear, desde el intérprete de comandos de "batman", la palabra **"ent_88k"**. Dicho programa, permite entregar los ficheros indicados anteriormente, así como consultar los resultados de la ejecución de un conjunto de pruebas utilizadas por el corrector.

Al entrar en el programa, éste pide la identificación del usuario. Tomaremos como identificación de usuario el número de matrícula de uno de los integrantes del grupo. El programa mostrará el mensaje:

Introduzca su identificador (Num. matricula): 990999

El usuario deberá introducir el número de matrícula de uno de los integrantes del grupo (p.e. 990999).

Si es la primera vez que el usuario entra en el sistema de entrega, el programa le invitará a que introduzca una palabra clave ("password") mostrando el siguiente mensaje:

Se va a establecer password.

Password:

El usuario deberá introducir una palabra clave (no se mostrará en pantalla). Para confirmar que no se ha producido ningún error al introducir el "password" se vuelve a pedir:

Repita el password tecleado anteriormente:

Si se ha producido algún error se reintentará establecer el password de nuevo.

Después de mostrar este mensaje el programa termina. Si el comando se ha ejecutado con éxito se mostrarán los datos que ha registrado el sistema de cada uno de los integrantes del grupo. Seguidamente aparecerá el siguiente mensaje:

SE HAN DADO DE ALTA LOS SIGUIENTES ALUMNOS:
123433342 990999 PEREZ PEREZ JESUS

Se ha asignado password al usuario 990999.
- NO LO OLVIDE
- NO LO APUNTE
- NO LO DIVULGUE

suponiendo que el grupo esté compuesto por un único alumno (Jesús Pérez Pérez con DNI n.º 123433342 y número de matrícula 990999) y la información que aparece en el fichero **autores** es:

990999 ; 123433342 ; PEREZ PEREZ JESUS; g990999@zipi.fi.upm.es

Si dicho alumno no aparece en las listas disponibles por el departamento o no ha introducido correctamente alguno de los datos, se mostrará un mensaje de error.

A continuación se mostrará el siguiente menú:

OPCIONES:
1. Mandar Ficheros.
2. Consultar Resultados.
3. Cancelar Entregas
4. Bloquear la Entrega
5. Ayuda !!!!
6. Noticias.
q Abandonar
>>>>

A continuación se explica cada una de las opciones del menú.

MANDAR FICHEROS

Esta opción permite enviar los ficheros de un práctica o proyecto, que deberán estar en el directorio de trabajo del usuario. Si el comando se ejecuta correctamente se mostrarán los siguientes mensajes:

MANDANDO EL FICHERO filtro.ens ...OK.

Si alguno de los ficheros no se encuentra, el programa lo comunicará al usuario. Por ejemplo:

MANDANDO EL FICHERO filtro.ens ...
No se puede abrir el fichero filtro.ens
Entrega cancelada.

El servidor de entregas intenta asegurar que cada uno de los ficheros tiene el formato correcto. En nuestro caso esto se traduce en que el fichero se va a poder ensamblar correctamente cuando se realice la corrección. Si el comando de ensamblado no ha finalizado con éxito se mostrará un mensaje:

EL FICHERO filtro.ens NO TIENE EL FORMATO CORRECTO
ENTREGA NO REALIZADA

En este caso el alumno deberá comprobar que se puede ensamblar correctamente el fichero y que contiene todas y cada una de las etiquetas que es obligatorio que aparezcan en dicho fichero.

En el caso de que se genere un error en la entrega de los ficheros, el programa de entrega termina su ejecución y se muestra el "prompt" del sistema operativo. Si se desea realizar una nueva entrega se volverá a teclear el comando.

CANCELAR ENTREGAS

Esta opción permite cancelar todas las entregas realizadas desde la última corrección. El identificador del grupo de alumnos será eliminado de la lista de proyectos pendientes de corregir. Si se han realizado varias entregas se cancelarán todas ellas.

CONSULTAR RESULTADOS

Esta opción permite consultar los resultados de la corrección de la entrega de un proyecto. El programa pide el nombre del fichero en el que se copiarán los resultados de la ejecución del conjunto de pruebas que componen el corrector. Se mostrará el siguiente mensaje:

La salida será redirigida a un fichero.
Nombre del fichero (ENTER para salida por pantalla) ?? result.txt

En este caso se grabarán los resultados de las pruebas en el fichero **result.txt**. Si como respuesta al mensaje se teclea ENTER, los resultados serán mostrados por pantalla. El nombre del fichero que se proporciona al programa (**result.txt**) no debe existir en el disco.

Esta opción se incluye para permitir la corrección automática de los proyectos. El alumno/grupo no debe utilizar este programa para depurar su código. Debe ser el propio alumno/grupo quien construya su conjunto de pruebas que le permita comprobar que el proyecto funciona correctamente.

BLOQUEO DE LA ENTREGA

Si el usuario se compromete a no entregar más veces el código del proyecto en la convocatoria en curso, puede bloquear la entrega para mayor seguridad. Si se ejecuta esta opción no se podrá volver a realizar una nueva entrega de los ficheros asociados al proyecto. Si el comando se ejecuta satisfactoriamente se mostrará el mensaje:

ENTREGA BLOQUEADA.

AYUDA

Esta opción mostrará en pantalla una breve descripción de cada una de las opciones del programa de entrega. No significa que se vaya a proporcionar ayuda para la realización del proyecto.

NOTICIAS

Esta opción es puramente informativa. Permite notificar al alumno modificaciones en la especificación del proyecto o, en general, noticias de interés de la asignatura asociada al mismo. El programa pide el nombre de fichero en el que se copiarán las noticias.

La salida será redirigida a un fichero.
Nombre del fichero (ENTER para salida por pantalla) ?? noticias.txt

En este caso se grabará la información relativa a la asignatura en el fichero **noticias.txt**. Si como respuesta al mensaje se teclea ENTER, la información será mostrada por pantalla.

ABANDONAR

Termina la ejecución del programa de entrega. Si se realiza con éxito se mostrará el mensaje:

Cerrando la conexion

y a continuación aparecerá el "prompt" del sistema operativo.

NOTA: no se corregirá ninguna entrega que no se atenga a estas normas y se considerará por lo tanto como no presentada.

Sistema de entrega en la red de PCs del Centro de Cálculo

Aquellos alumnos que desarrollen el proyecto sin utilizar los medios del Centro de Cálculo (utilización de PCs particulares) pueden realizar la entrega de ficheros desde la red de PCs de la Facultad. Para ello se deben situar en la unidad de trabajo donde tengan los ficheros del proyecto y teclear lo siguiente (suponiendo que los ficheros se encuentran

en una carpeta denominada "PROYECTO" dentro de un pen-drive o lápiz de datos que se ubica en la unidad "H:" de Windows):

```
G:\>h:
H:\>cd proyecto
H:\PROYECTO>g:\datsi\entregas\ent_88k
```

El diálogo que este programa ofrece al usuario es el mismo que el que se ejecuta en *batman* (véase la sección anterior).

Los alumnos que utilicen este mecanismo de entrega del proyecto deberán tomar las debidas precauciones para evitar la difusión o copia de su código. ¡Se recomienda trabajar siempre sobre un dispositivo extraíble (*pendrive*) y nunca sobre disco duro!

Si no se tienen en cuenta estas precauciones, los ficheros podrían ser leídos por otros usuarios y por lo tanto **COPIADOS**, con las consecuencias que se derivan de las normas de la asignatura.

Sistema de entrega a través del servidor web del DATSI

Las mismas operaciones que permite realizar la aplicación `ENT_88k` descrita en el apartado anterior pueden ser efectuadas mediante un navegador Web, para lo que será necesario conectarse a la URL: "http://www.datsi.fi.upm.es/Practicas".

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:

  $C:\mathit{Facultad}\backslash\mathit{Ensamblador}$
- 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:

  $cd\ \ C:\mathit{Facultad}\backslash\mathit{Ensamblador}\backslash 88k\_Windows$

  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".