Práctica de Programación en Ensamblador (88110) Laboratorio de Estructura de Computadores

Departamento de Arquitectura y Tecnología de Sistemas Informáticos

desde Feb-2015

La práctica consiste en la programación en ensamblador del Motorola 88110 de un conjunto de rutinas que permitan realizar determinadas operaciones en una pequeña tabla que simula el comportamiento de la FAT (tabla de asignación de ficheros) del sistema operativo MS-DOS.

AVISO

El enunciado de esta práctica coincide con el correspondiente cursos anteriores, por lo que aquellos alumnos que tengan que repetir o corregir la práctica podrán utilizar los programas ya realizados por ellos mismos durante dichos cursos académicos.

La práctica se realizará de forma individual.

Sistema operativo MS-DOS

Este sistema emplea disquetes y unidades de disco, pero en todos los casos los sectores son de 512 bytes. Las unidades de disco pueden dividirse en particiones, considerando cada partición como una unidad de disco independiente. La unidad de información en que se almacenan los ficheros es la agrupación (*cluster*), que es un conjunto de n sectores contiguos ($n = 2^k, k = 0, 1, 2, ..$).

Un disco (una partición) se divide en cuatro zonas como puede verse en la figura 1. Estas zonas son:

Figura 1. Estructuración de un disco en MS-DOS.

1. **Boot o sector de arranque**. Es el primero del disco y contiene un pequeño programa que busca los ficheros ocultos IO.SYS y MSDOS.SYS en el directorio raíz, si los encuentra carga el sistema operativo y en caso contrario muestra un mensaje de error. Además, contiene información como el nombre y versión del sistema operativo, características del disco e información sobre la FAT y el directorio raíz.

2. **Copias de la FAT**. La FAT es la tabla de asignación de ficheros (*File Allocate Table*) que se describirá posteriormente. En MS-DOS la FAT se encuentra duplicada para hacer el sistema más tolerante a fallos de funcionamiento.

3. **Directorio raíz**. Es la zona del disco en la que se almacenan los directorios y determinados ficheros. Tiene un tamaño prefijado, por ejemplo, un disco de 244 Mbytes tiene un directorio raíz de 32 sectores, que suponen 512 entradas o alojamientos para directorios o ficheros. Cada entrada o alojamiento de un directorio, subdirectorio o fichero tiene una extensión de 32 bytes y consta de los campos que pueden verse en la figura 2 y que son:

Figura 2. Estructura de cada alojamiento del directorio raíz.

- **Nombre**. Tiene una extensión de 8 bytes. Si el primer byte de este campo es `H'00` quiere decir que ese alojamiento no ha sido utilizado, y si es `H'E5` indica que el alojamiento fue utilizado y posteriormente borrado. Para añadir un fichero o directorio nuevo se busca el primer alojamiento que tenga su primer byte a `H'00` ó a `H'E5`.
- **Extensión**. Ocupa 3 bytes, con lo cual los ficheros y directorios quedan denominados con un nombre de ocho caracteres y una extensión de tres.
- **Atributos**. Es un byte que indica ciertas características de un alojamiento: fichero de sólo lectura (R), fichero oculto (H), fichero de sistema (S), fichero a archivar (A), subdirectorio (D).
- **Hora y fecha**. Se representa la hora, minutos y segundos de la última actualización del fichero. Análogamente se representa el año, mes y día.
- **Primera agrupación**. Este campo sirve para determinar el primer sector del disco en que se encuentra el fichero, al describir la FAT se analizará cómo se realiza.
- **Tamaño**. Este campo indica el tamaño del fichero en bytes.

4. **Datos y directorios**. Esta zona corresponde al resto del disco y es donde se almacenan los ficheros y los subdirectorios, de forma que su tamaño y ubicación están regulados por la FAT.

Tabla de asignación de ficheros (FAT)

En el sistema operativo MS-DOS existen tablas de asignación de 12, 16 y 32 bits. Nos referiremos exclusivamente a la de 16 bits ya que el tratamiento sería el mismo con otro número de bits. En este caso, cada elemento de la FAT tiene 16 bits y van numerados desde 0 hasta 65535 $= 2^{16} - 1 =$`H'FFFF`, indicando la agrupación que ocupa en el disco. Los dos primeros elementos no especifican ficheros o subdirectorios, de forma que:

- El primero especifica el formato del disco, indicando el tipo de disco que tenemos.
- El segundo está reservado para otras funciones del sistema operativo, siendo su valor estándar `H'FFFF`.

El resto de elementos identifican una agrupación (desde la 2 hasta la 65535) de forma que su contenido tiene el siguiente significado:

- `H'0000` indica que se trata de una agrupación disponible
- Desde `H'FFF8` hasta `H'FFFF` indican que es la última agrupación de un fichero o de un subdirectorio

- `H'FFF7` indica que es una agrupación con sectores dañados
- Cualquier otro valor, `H'XXXX`, indica que es una agrupación usada en un fichero, siendo el valor almacenado `H'XXXX` la agrupación donde continúa el fichero

Simulador de la FAT

En esta práctica, manteniendo todas las características conceptuales vistas en el sistema MS-DOS, se considerará una FAT de 8 bits, lo que supone una FAT con un número máximo de 256 entradas. Se considerará que las cuatro primeras entradas están reservadas y que el contenido de las restantes tiene el siguiente significado:

- `H'00` indica una agrupación disponible
- `H'FF` indica que es la última agrupación de un fichero o de un subdirectorio
- `H'XX` indica que es una agrupación usada en un fichero siendo `H'XX` la agrupación donde continúa el fichero

La unidad de disco considerada tiene sectores de 64 bytes y las agrupaciones son de un sector. El directorio raíz consta de dos sectores y cada alojamiento para un directorio, subdirectorio o fichero consta de:

- 4 bytes para el nombre. El nombre puede tener como máximo cuatro caracteres, si tuviese menos el nombre termina cuando se encuentre el carácter NUL (`H'00`).
- 1 byte para el atributo. Solamente puede haber dos atributos, `H'41 (A)` para indicar que se trata de un archivo[1] y `H'44 (D)` para indicar que se trata de un directorio.
- 1 byte para la primera agrupación.
- 2 bytes para el tamaño, expresado en bytes.

por lo tanto, cada sector tiene 8 alojamientos y el directorio raíz 16.

La FAT que se considera en la simulación que se hace de la unidad de disco está contenida en un sector, por lo tanto, de las 256 entradas posibles solo se contemplan 64, desde la 0 (`H'00`) hasta la 63 (`H'3F`). Como las cuatro primeras entradas quedan reservadas, la zona de datos del disco está constituida por 60 sectores, quedando el disco que se simula como puede verse en la figura 3.

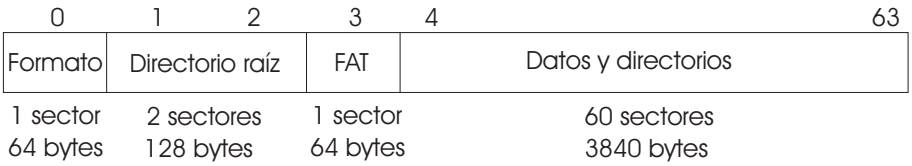

Figura 3. Estructura de la unidad de disco simulada.

[1] Denominaremos 'Archivo' a un fichero que no es un subdirectorio, 'Directorio' al que sí lo es y utilizaremos 'Fichero' como nombre genérico para denominar a ambos.

En esta unidad de disco que se simula, el valor de cada entrada de la FAT coincide con el valor del sector absoluto donde se almacena el fichero.

Por último, en la implementación de la FAT simulada y las rutinas que han de construirse para gestionarla, se utilizarán dos tipos de nombre de fichero:

Nombre relativo a un subdirectorio: Contendrá un máximo de cuatro caracteres alfanuméricos comenzando por un carácter alfabético. Ejemplo: "`PROB`". Si el nombre tuviera menos de cuatro caracteres, tendría que finalizar con el carácter terminador `H'00`. Ejemplo: "`Z\0`".

Ruta o nombre completo del fichero: Incluirá la relación de subdirectorios que permiten llegar a dicho fichero desde el directorio raíz, separados por un carácter especial, el separador '/' (`H'2F`). Ejemplo: "`/DIR2/SD22/SLAB/PROB\0`", donde el carácter '`\0`' (`H'00`) se utiliza como terminador de la cadena de caracteres.

Ejemplo

A continuación se describe el contenido de un disco con una serie de ficheros y directorios que se utilizará como ejemplo. El contenido del disco en un instante dado es el que se muestra en la figura 4.

Las estructuras de datos que lo describen son las que se expresan a continuación.

El contenido de la FAT es:

Sector	Contenido	Sector	Contenido	Sector	Contenido	Sector	Contenido
0	0xFF	16	0xFF	32	0xFF	48	0xFF
1	0xFF	17	0xFF	33	0x00	49	0xFF
2	0xFF	18	0xFF	34	0x00	50	0x33
3	0xFF	19	0x00	35	0x00	51	0xFF
4	0xFF	20	0x1D	36	0x00	52	0xFF
5	0xFF	21	0x00	37	0x2A	53	0xFF
6	0x07	22	0xFF	38	0xFF	54	0x38
7	0xFF	23	0xFF	39	0xFF	55	0xFF
8	0xFF	24	0xFF	40	0x30	56	0xFF
9	0xFF	25	0xFF	41	0x00	57	0x00
10	0x12	26	0x00	42	0xFF	58	0x00
11	0x00	27	0x00	43	0x00	59	0x00
12	0x00	28	0x00	44	0x2D	60	0x00
13	0xFF	29	0x25	45	0x31	61	0x00
14	0x14	30	0x00	46	0x00	62	0x00
15	0xFF	31	0x2C	47	0x00	63	0x00

Las entradas del directorio raíz son:

Entrada	Contenido
0	44495231 44040000 (DIR1 D 04 0000)
1	44495232 44050000 (DIR2 D 05 0000)
2	494F0000 41060064 (IO A 06 0100)
3	444F5300 41080032 (DOS A 08 0050)

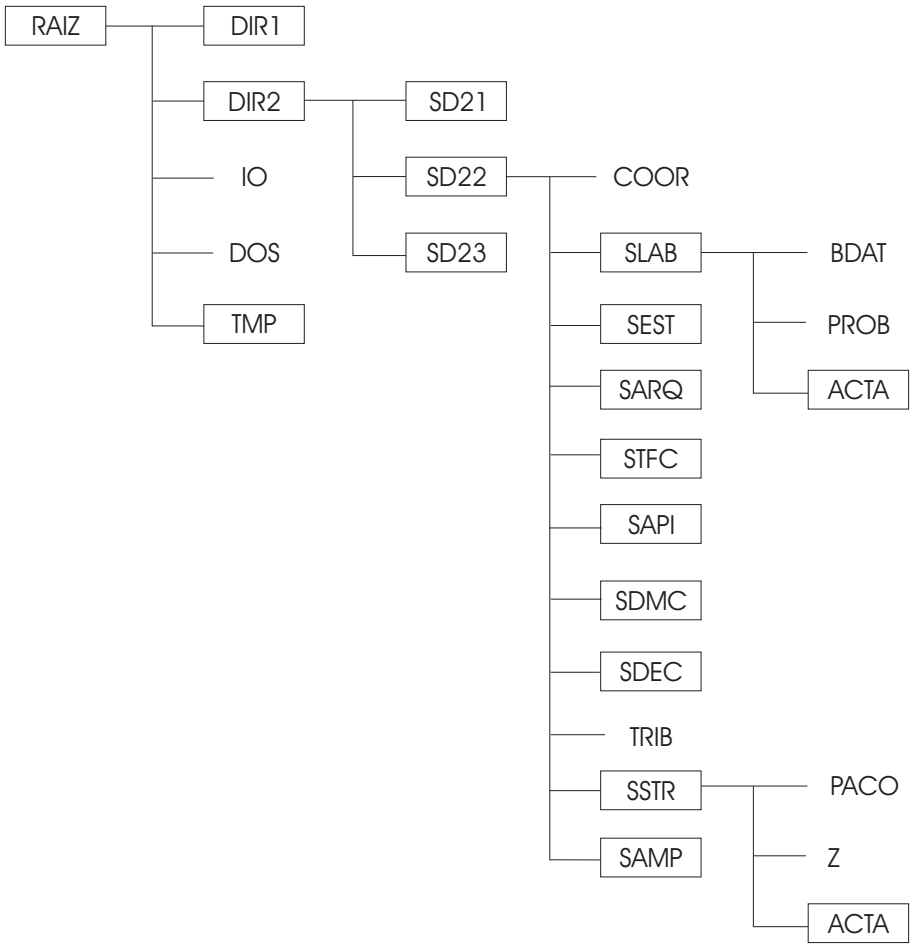

Figura 4. Directorio en árbol del ejemplo considerado.

```
   4     E5495233 44090000 (entrada borrada)
   5     544D5000 44200000 ( TMP D 32 0000)
   6     00000000 00000000 (entrada libre)
   ..............................
```

El sector 5 del disco corresponde al directorio DIR2, que contiene:

```
Entrada  Contenido
   0     53443231 44090000 (SD21 D 09 0000)
   1     53443232 440A0000 (SD22 D 10 0000)
   2     53443233 440D0000 (SD23 D 13 0000)
   3     00000000 00000000 (entrada libre)
   ..............................
```

El sector 10 del disco corresponde al subdirectorio SD22, que contiene:

```
Entrada  Contenido
   0     434F4F52 410E012C (COOR A 14 0300)
   1     534C4142 440F0000 (SLAB D 15 0000)
```

```
2     53455354 44100000 (SEST D 16 0000)
3     53415251 44110000 (SARQ D 17 0000)
4     53544643 44160000 (STFC D 22 0000)
5     53415049 44170000 (SAPI D 23 0000)
6     53444D43 44180000 (SDMC D 24 0000)
7     53444543 44190000 (SDEC D 25 0000)
```

El subdirectorio SD22 continúa en el sector 18, tal como indica la entrada 10 de la FAT. Este sector contiene:

```
Entrada  Contenido
   0     54524942 411F00FA (TRIB A 31 0250)
   1     E5495233 44090000 (entrada borrada)
   2     53535452 44260000 (SSTR D 38 0000)
   3     53414D50 44270000 (SAMP D 39 0000)
   4     00000000 00000000 (entrada libre)
   ..............................
```

El subdirectorio SLAB ocupa el sector 15 del disco y contiene:

```
Entrada  Contenido
   0     42444154 41280078 (BDAT A 40 0120)
   1     50524F42 41320055 (PROB A 50 0085)
   2     41435441 44350000 (ACTA D 53 0000)
   3     00000000 00000000 (entrada libre)
   ..............................
```

El subdirectorio SSTR ocupa el sector 38 del disco y contiene:

```
Entrada  Contenido
   0     5041434F 41340032 (PACO A 52 0050)
   1     E5495233 44290000 (entrada borrada)
   2     5A000000 4136006A (Z    A 54 0106)
   3     41435441 44370000 (ACTA D 55 0000)
   ..............................
```

Descripción de los programas de la práctica

La práctica consistirá en implementar ocho subrutinas, que se relacionan tal y como se indica en la figura 5. Además deberá implementarse un programa principal que realizará las pruebas de funcionamiento necesarias. Este programa será automáticamente sustituido por el que determine el Departamento durante las pruebas de evaluación automática de la práctica. Se ha preferido partir el programa en un número alto de subrutinas para facilitar su depuración, ya que así se trabajará con fragmentos de código pequeños, que por tanto serán manejables. Además, al tener el programa segmentado en varias partes, el corrector automático proporcionará información más precisa sobre qué partes están bien o están mal. Aunque se haya realizado esta división en fragmentos, si los miembros del grupo consideran conveniente desarrollar otras rutinas que sirvan como base para las que específicamente se tienen que construir, no sólo pueden hacerlo, sino que se trata de una práctica recomendable. Por ejemplo, se podría implementar una subrutina para localizar la primera entrada libre de la FAT, u otra para comprobar si dos nombres son iguales.

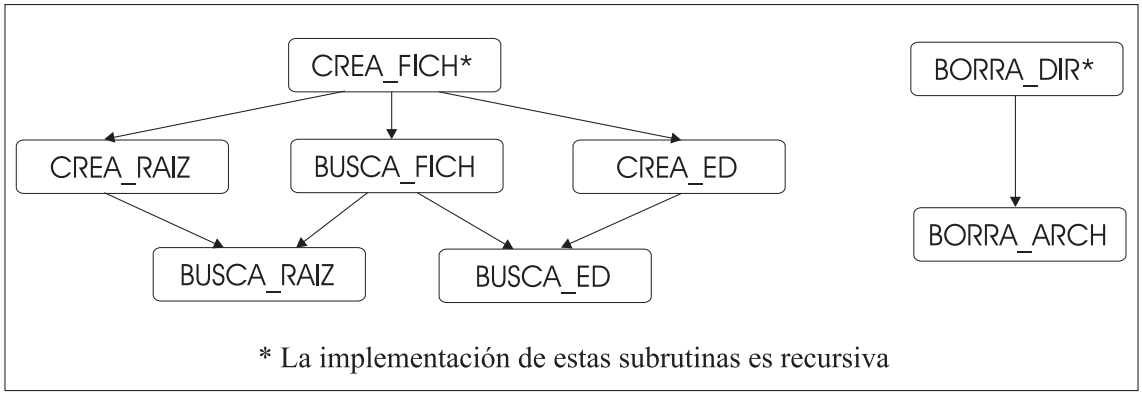

Figura 5. Jerarquía de las rutinas de la práctica.

Programa Principal

Se construirá un programa principal para probar cada una de las subrutinas de la práctica. Dicho programa principal se encargará de inicializar el puntero de pila `r30`, pasar los parámetros de la subrutina en la pila de usuario e invocar a dicha subrutina.

Busca un fichero en el directorio raíz

```
DirEnt = BUSCA_RAIZ (DirDisco, NombreFich)
```

Parámetros:

- **DirDisco:** Es la dirección de memoria del 88110 en que comienza el disco simulado. A partir de esa dirección se encuentran las sucesivas agrupaciones que componen el disco.
- **NombreFich:** Es una dirección de memoria en la que comienza un nombre de fichero válido que no incluye la ruta de acceso al mismo, es decir, se compone de cuatro caracteres o bien de entre uno y tres caracteres seguidos por un terminador (`H'00`) o por un separador (`H'2F`).

Los dos parámetros de esta función se pasan en la pila.

Resultado:

- **DirEnt:** Es la dirección de memoria en la que se encuentra la entrada del directorio raíz que corresponde al fichero (archivo o directorio) especificado. En caso de no encontrarse dicho fichero en ninguna de las dos agrupaciones del directorio raíz, la rutina devolverá un resultado igual a -1. El resultado se devuelve por valor en el registro `r29`.

Descripción:

La rutina `BUSCA_RAIZ` tratará de localizar el fichero especificado como parámetro en alguna de las entradas del directorio raíz y devolver, en caso de finalización con éxito, la dirección de memoria donde comienza dicha entrada en el directorio raíz. Si se produce algún error, se devolverá el valor -1 como retorno. Puesto que el directorio raíz no es ampliable, únicamente hay que comprobar si se encuentra el fichero especificado en alguna de las agrupaciones predeterminadas (en S1 ó en S2).

La rutina `BUSCA_RAIZ` consta de los siguientes pasos:

1. Inicia a cero una variable local `v_nombre` de tamaño igual a una palabra, en la que copiará los caracteres del nombre especificado en el segundo parámetro. La copia se realizará desde el primero de dichos caracteres hasta que se encuentre un separador (`H'2F`) o un terminador (`H'00`) (que no se incluyen) o hasta que se hayan copiado cuatro caracteres.

2. Localiza, a partir de la dirección facilitada como primer parámetro, la dirección de memoria que corresponde a la primera de las dos agrupaciones que componen el directorio raíz.

3. Recorre cada una de las entradas de las dos agrupaciones consecutivas localizadas en el paso anterior, comparando la componente de dicha entrada que contiene el nombre del fichero con el nombre almacenado en `v_nombre`.

4. Si para alguna de las entradas de cualquiera de las dos agrupaciones se encuentra que el nombre completo de la entrada coincide con el nombre almacenado en `v_nombre`, se procede a copiar la dirección de dicha entrada sobre el registro `r29` y se da por finalizada la subrutina, devolviendo el control al programa llamante. Si tras recorrer todas las entradas de las dos agrupaciones no se localiza el nombre recogido como parámetro, se devuelve el control al llamante con el valor -1 en `r29`.

Busca un fichero en un directorio especificado

```
DirEnt = BUSCA_ED (DirDisco, NombreFich, DirDirec)
```

Parámetros:

- **DirDisco:** Es la dirección de memoria del 88110 en que comienza el disco simulado. A partir de esa dirección se encuentran las sucesivas agrupaciones que componen el disco.
- **NombreFich:** Es una dirección de memoria en la que comienza un nombre de fichero válido que no incluye la ruta de acceso al mismo, es decir, se compone de cuatro caracteres o bien de entre uno y tres caracteres seguidos por un terminador (`H'00`) o por un separador (`H'2F`).
- **DirDirec:** Es la dirección de memoria en la que se encuentra la entrada de directorio que describe el directorio en que se desea realizar la búsqueda. En esta dirección se encontrará por tanto el nombre del directorio en que se realizará la búsqueda, seguido de la etiqueta que lo identifica como directorio ('D'), seguido del número de agrupación en que comienza, y finalizado por un campo en que se indica la longitud del fichero (en el caso del directorio será igual a cero).

Los tres parámetros de esta función se pasan en la pila.

Resultado:

- **DirEnt:** Es la dirección de memoria en la que se encuentra la entrada del directorio especificado como tercer parámetro que corresponde al fichero (archivo o directorio) que se busca. En caso de no encontrarse dicho fichero en ninguna de las agrupaciones de que consta el directorio especificado, la rutina devolverá un resultado igual a -1. El resultado se devuelve por valor en el registro `r29`.

Descripción:

La rutina `BUSCA_ED` tratará de localizar el archivo especificado como segundo parámetro en alguna de las entradas del directorio que se especifica en el tercer parámetro. Si la localización tiene éxito, la rutina devolverá la dirección de memoria donde comienza dicha entrada en el directorio correspondiente. Si se produce algún error, se devolverá el valor -1 como retorno. Para determinar que no se ha encontrado el fichero, será necesario haber comprobado que no se encuentra en ninguna de las agrupaciones de que consta el directorio especificado.

La rutina `BUSCA_ED` consta de los siguientes pasos:

1. Inicia a cero una variable local `v_nombre` de tamaño igual a una palabra, en la que copiará los caracteres del nombre especificado en el segundo parámetro. La copia se realizará desde el primero de dichos caracteres hasta que se encuentre un separador (`H'2F`) o un terminador (`H'00`) (que no se incluyen) o hasta que se hayan copiado cuatro caracteres.

2. Lee, a partir de la dirección facilitada como tercer parámetro, el número de agrupación en que comienza el directorio sobre el que se realizará la búsqueda.

3. Localiza, a partir de la dirección facilitada como primer parámetro y el número de agrupación obtenido en el paso anterior, la dirección de memoria que corresponde a la primera agrupación de las que componen el directorio especificado.

4. Recorre cada una de las entradas de la agrupación localizada, comparando la componente de dicha entrada que contiene el nombre del fichero con el nombre almacenado en la variable `v_nombre`.

5. Si para alguna de las entradas de la agrupación se encuentra que el nombre completo de la entrada coincide con el nombre almacenado en `v_nombre`, se procede a copiar la dirección de dicha entrada sobre el registro `r29` y se da por finalizada la subrutina, devolviendo el control al programa llamante. Si tras recorrer todas las entradas de la agrupación no se localiza el nombre recogido como parámetro, se continúa en el siguiente paso

6. Lee la entrada de la FAT correspondiente a la agrupación cuyo recorrido acaba de finalizar. Si el contenido de dicha entrada corresponde a la marca de 'final de fichero' (`H'FF`), la búsqueda habrá terminado, por lo que se cargará el registro `r29` con el valor -1 y se devolverá el control al programa llamante. En caso contrario, continuará por en el siguiente paso.

7. Localiza, a partir de la dirección facilitada como primer parámetro y el número de agrupación obtenido de la FAT en el paso anterior, la dirección de memoria que corresponde a la siguiente agrupación de las que componen el directorio especificado, continuando la subrutina por el paso 4.

Busca un fichero a partir de su nombre completo (ruta)

```
DirEnt = BUSCA_FICH (DirDisco, NombreRuta)
```

Parámetros:

- **DirDisco:** Es la dirección de memoria del 88110 en que comienza el disco simulado. A partir de esa dirección se encuentran las sucesivas agrupaciones que componen el disco.

- **NombreRuta:** Es una dirección de memoria en la que está almacenado un nombre completo de fichero, es decir un nombre que incluye la ruta de acceso al mismo. Consecuentemente estará formado por un separador (opcional) seguido de un conjunto de nombres de directorios seguidos de separadores, y finalizado por un nombre válido de fichero (archivo o directorio).

Los dos parámetros de esta función se pasan en la pila.

Resultado:

- **DirEnt:** Es la dirección de memoria en la que se encuentra la entrada del directorio que corresponde al fichero (archivo o directorio) que se busca. En caso de no encontrarse dicho fichero o alguno de los directorios que componen la ruta, la rutina devolverá un resultado igual a -1. El resultado se devuelve por valor en el registro `r29`.

Descripción:

La rutina `BUSCA_FICH` tratará de localizar el fichero cuya ruta completa está especificada como segundo parámetro. Si la localización tiene éxito, la rutina devolverá la dirección de memoria donde comienza la entrada que describe el fichero especificado en el directorio en que se encuentra. Si se produce algún error, se devolverá el valor -1 como retorno. Para determinar que no se ha encontrado el fichero, será necesario haber comprobado que no se encuentra en ninguna de las agrupaciones de que consta el directorio especificado o bien que no existe alguno de los directorios que forman parte de la ruta completa del fichero.

La rutina `BUSCA_FICH` se implementará de acuerdo a los siguientes pasos:

1. Carga un registro `r_dir` con la dirección facilitada como segundo parámetro de esta función. Si dicho registro está apuntando a un carácter separador (`H'2F`), se incrementará en una unidad.

2. Localiza, mediante una llamada a la subrutina `BUSCA_RAIZ`, la dirección de la entrada correspondiente al fichero de primer nivel de la ruta especificada. Para ello, llama a `BUSCA_RAIZ`, pasándole como parámetros la dirección del disco y la dirección anotada en `r_dir` en el paso anterior. Si `BUSCA_RAIZ` devuelve -1, se trata de un error, por lo que se devolverá el control al programa llamante.

3. Actualiza el registro `r_dir` de modo que quede apuntando al primer carácter tras el siguiente separador (que corresponderá al comienzo del nombre de fichero del siguiente nivel al ya tratado). Si en lugar de un separador se localiza el terminador de la cadena de caracteres, o si tras el separador se encuentra el terminador, querrá decir que el nombre que se trata de buscar en este paso es el del último componente de la ruta. Si en este recorrido se detectan más de cuatro caracteres antes de llegar al separador o al terminador, se tratará de un error, por lo que se devolverá el control al llamante con -1 en `r29`.

4. Llama a la subrutina `BUSCA_ED` pasándole como parámetro la dirección de comienzo del nombre obtenido en el paso anterior (almacenada en `r_dir`). Dicha subrutina devolverá la dirección de la entrada correspondiente al archivo o al directorio del nivel en curso, o bien el valor -1. En este último caso, se tratará de un error, por lo que se devolverá el control al programa llamante. En caso contrario:

   a) Si el fichero tratado es el último componente de la ruta especificada, se copiará sobre `r29` la dirección de la entrada de directorio que describe ese fichero y se devolverá el control al programa llamante.
   b) Si el fichero tratado no es el último componente de la ruta especificada y es de tipo '`D`', se procederá de nuevo con el paso 3.
   c) Si el fichero no es el último componente de la ruta especificada y es de tipo '`A`', se devolverá el control al llamante, con -1 en `r29`.

Crea un fichero en el directorio raíz

```
DirEnt = CREA_RAIZ (DirDisco, NombreFich, TipoFich)
```

Parámetros:

- **DirDisco:** Es la dirección de memoria del 88110 en que comienza el disco simulado. A partir de esa dirección se encuentran las sucesivas agrupaciones que componen el disco.
- **NombreFich:** Es una dirección de memoria en la que comienza un nombre de fichero válido que no incluye la ruta de acceso al mismo, es decir, se compone de cuatro caracteres o bien de entre uno y tres caracteres seguidos por un terminador (`H'00`) o por un separador (`H'2F`).
- **TipoFich:** Es una palabra cuyo carácter menos significativo indica el tipo del fichero a crear: '`A`' para seleccionar un archivo, o '`D`' para seleccionar un directorio. Se pasa por valor.

Los tres parámetros de esta función se pasan en la pila.

Resultado:

- **DirEnt:** Es la dirección de memoria en la que se encuentra la entrada de directorio que corresponde al fichero (archivo o directorio) que se ha creado. En caso de haberse producido algún error durante la creación del fichero, la rutina devolverá un resultado igual a -1. El resultado se devuelve por valor en el registro `r29`.

Descripción:

La rutina `CREA_RAIZ` tratará de crear en el directorio raíz el fichero especificado en los parámetros segundo y tercero. Si la creación se realiza correctamente, la rutina devolverá la dirección de memoria donde comienza la entrada que describe el fichero especificado en el directorio raíz. Si se produce algún error (p.e. el fichero ya existe, o no hay espacio disponible en el disco), se devolverá el valor -1 como retorno. Para determinar si hay entradas libres en el directorio raíz para crear el fichero solicitado, puede ser necesario recorrer las dos agrupaciones que componen este directorio.

La rutina `CREA_RAIZ` se implementará de acuerdo a los siguientes pasos:

1. Llama a la subrutina `BUSCA_RAIZ`, pasándole como parámetros la dirección del disco y el nombre del fichero a buscar. Si `BUSCA_RAIZ` devuelve un valor distinto a -1 es porque el fichero ya existe, por lo que no es posible crearlo: se almacena -1 en `r29` y se devuelve el control al programa llamante.
2. Localiza la primera entrada libre del directorio raíz, que es la que se utilizará para ubicar el fichero que se ha de crear. Si no hay ninguna entrada libre, se almacena -1 en `r29` y se devuelve el control al programa llamante.

3. Recorre la FAT para localizar la primera agrupación libre del disco (la primera cuya entrada de la FAT tenga el valor `H'00`). Si el disco está lleno (no hay agrupaciones disponibles), se devuelve el control al programa llamante con un valor -1 en `r29`. En caso contrario, se escribirá el valor `H'FF` en la entrada de la FAT cuyo valor era `H'00`. Además, si el fichero que se está creando es un directorio, se inicializará a `H'00` la agrupación que se le ha asignado.

4. Copia el nombre del fichero especificado en el segundo parámetro sobre la entrada disponible que se localizó en el paso 2. La copia se hace desde su primer carácter hasta detectar un separador o el terminador (que no se copian), o bien hasta que se hayan copiado 4 caracteres. Si el nombre ocupa menos de 4 bytes, se completa con bytes a cero. A continuación se copiará el carácter que indica el tipo de fichero a crear (el que se ha pasado en el tercer parámetro de esta subrutina). El byte siguiente lo ocupará el número de agrupación asignada inicialmente al fichero que se está creando (el que se ha obtenido en el paso 3). Finalmente se escribirán dos bytes a cero, indicando que el fichero se crea con una longitud inicial de 0 bytes. Con esta escritura queda completa la entrada del directorio correspondiente al fichero recién creado.

5. Devuelve el control al programa llamante, una vez haya copiado en `r29` la dirección obtenida en el paso 2.

Crea un fichero en un directorio especificado

```
DirEnt = CREA_ED (DirDisco, NombreFich, DirDirec, TipoFich)
```

Parámetros:

- **DirDisco:** Es la dirección de memoria del 88110 en que comienza el disco simulado. A partir de esa dirección se encuentran las sucesivas agrupaciones que componen el disco.

- **NombreFich:** Es una dirección de memoria en la que comienza un nombre de fichero válido que no incluye la ruta de acceso al mismo, es decir, se compone de cuatro caracteres o bien de entre uno y tres caracteres seguidos por un terminador (`H'00`) o por un separador (`H'2F`).

- **DirDirec:** Es la dirección de memoria en la que se encuentra la entrada de directorio que describe el directorio en que se desea crear el fichero. A partir de esta dirección se encontrará por tanto el nombre del directorio en que se creará el nuevo fichero, seguido de la etiqueta que lo identifica como directorio ('D'), del número de agrupación en que comienza, y del campo que indica la longitud, que tendrá el valor 0.

- **TipoFich:** Es una palabra cuyo carácter menos significativo indica el tipo del fichero a crear: '`A`' para seleccionar un archivo, o '`D`' para seleccionar un directorio. Se pasa por valor.

Los cuatro parámetros de esta función se pasan en la pila.

Resultado:

- **DirEnt:** Es la dirección de memoria en la que se encuentra la entrada de directorio que corresponde al fichero (archivo o directorio) que se acaba de crear. En caso de producirse algún error durante la creación del fichero, la rutina devolverá un resultado igual a -1. El resultado se devuelve por valor en el registro `r29`.

Descripción:

La rutina `CREA_ED` tratará de crear el fichero especificado como segundo parámetro en el directorio que se especifica en el tercer parámetro. Si la creación tiene éxito, la rutina devolverá la dirección de memoria donde comienza la entrada del fichero recién creado en el directorio correspondiente. Si se produce algún error, por ejemplo que el fichero que se quiere crear ya exista, se devolverá el valor -1 como retorno.

La rutina `CREA_ED` consta de los siguientes pasos:

1. Llama a la subrutina `BUSCA_ED`, pasándole como parámetros la dirección del disco, la dirección de la entrada de directorio en que se ha de buscar y el nombre del fichero a buscar. Si `BUSCA_ED` devuelve un valor distinto a -1 es porque el fichero ya existe, por lo que no es posible crearlo: se almacena -1 en `r29` y se devuelve el control al programa llamante.

2. Localiza la primera entrada libre del directorio especificado, que es la que se utilizará para ubicar el fichero que se ha de crear:

   a) Si hay una entrada libre en las agrupaciones de las que consta el directorio, se empleará esta agrupación para almacenar la descripción del nuevo fichero a crear. El proceso continuará por el paso 3.

   b) Si no hay ninguna entrada libre en las agrupaciones de las que consta el directorio, habrá que ampliarlo, para lo que:

      1) Se recorre la FAT para localizar la primera agrupación libre del disco (`n_ag_libre`, la primera cuya entrada de la FAT tenga el valor `H'00`). Si el disco está lleno, se devuelve el control al programa llamante con un valor -1 en `r29`.
      2) Se inicializarán `H'00` todos los bytes de la agrupación libre que se acaba de localizar (la numerada como `n_ag_libre`).
      3) Se escribirá el número `n_ag_libre` en la entrada de la FAT que identificaba la última agrupación ocupada por el directorio que se está ampliando, que tenía el valor `H'FF`.
      4) Se escribirá el valor `H'FF` en la entrada `n_ag_libre` de la FAT, cuyo valor anterior era `H'00`. Se empleará esta nueva agrupación para hacer referencia al fichero que se está creando.

3. Se recorre la FAT para localizar la primera agrupación libre del disco (la primera cuya entrada de la FAT tenga el valor `H'00`). Si el disco está lleno, se devuelve el control al programa llamante con un valor -1 en `r29`. En caso contrario, se escribirá el

valor `H'FF` en la entrada de la FAT cuyo valor era `H'00` y, sólo en el caso de que se esté creando un directorio, se inicializará a `H'00` la agrupación seleccionada.

4. Copia el nombre del fichero especificado en el segundo parámetro sobre la entrada disponible que se localizó en el paso 2. La copia se hace desde su primer carácter hasta detectar un separador o el terminador (que no se copian), o bien hasta que se hayan copiado 4 caracteres. Si el nombre ocupa menos de 4 bytes, se completa con bytes a cero. A continuación se copiará el carácter que indica el tipo de fichero a crear (el que se ha pasado en el último parámetro a esta subrutina). El byte siguiente lo ocupará el número de agrupación asignada inicialmente al fichero que se está creando (el que se ha obtenido en el paso 3). Finalmente se escribirán dos bytes a cero, indicando que el fichero se crea con una longitud inicial de 0 bytes. Con esta escritura queda completa la entrada del directorio correspondiente al fichero recién creado.

5. Devuelve el control al programa llamante, una vez haya copiado en `r29` la dirección obtenida en el paso 2.

Crea un fichero cuyo nombre completo se especifica

```
DirEnt = CREA_FICH (DirDisco, NombreRuta, TipoFich)
```

Parámetros:

- **DirDisco:** Es la dirección de memoria del 88110 en que comienza el disco simulado. A partir de esa dirección se encuentran las sucesivas agrupaciones que componen el disco.

- **NombreRuta:** Es una dirección de memoria en la que está almacenado un nombre completo de fichero, es decir un nombre que incluye la ruta de acceso al mismo. Consecuentemente estará formado por un separador (opcional) seguido de un conjunto de nombres de directorios seguidos de separadores, y finalizado por un nombre válido de fichero (archivo o directorio).

- **TipoFich:** Es una palabra cuyo carácter menos significativo indica el tipo del fichero a crear: '`A`' para seleccionar un archivo, o '`D`' para seleccionar un directorio. Se pasa por valor.

Los tres parámetros de esta función se pasan en la pila.

Resultado:

- **DirEnt:** Es la dirección de memoria en la que se encuentra la entrada del directorio que corresponde al fichero (archivo o directorio) que se solicita crear. En caso de producirse un error al crear dicho fichero o alguno de los directorios que forman parte de la ruta, la rutina devolverá un resultado igual a -1. El resultado se devuelve por valor en el registro `r29`.

Descripción:

La rutina `CREA_FICH` tratará de crear en el disco el fichero cuya ruta completa está especificada como segundo parámetro y cuyo tipo se especifica en el tercer parámetro de la función. Si la creación tiene éxito, la rutina devolverá la dirección de memoria donde comienza la entrada del fichero recién creado en el directorio correspondiente. Si se produce algún error, por ejemplo que el fichero que se quiere crear ya exista y sea de distinto tipo al especificado en el tercer parámetro, se devolverá el valor -1 como retorno.

La rutina `CREA_FICH` se implementará mediante un algoritmo recursivo, hará uso del marco de pila y se desarrollará según los pasos indicados a continuación:

1. Llama a la subrutina `BUSCA_FICH`, pasándole como parámetros la dirección del disco y el nombre completo del fichero a buscar (el que se quiere crear). Si `BUSCA_FICH` devuelve un valor igual a -1 es porque el fichero no existe y la subrutina continuará por el paso 2. En caso contrario el fichero ya existe:

   a) Si es del mismo tipo que el especificado en el tercer parámetro de `CREA_FICH`, se devuelve el control al programa llamante, sin modificar el valor de `r29` recibido de `BUSCA_FICH`.

   b) Si es de distinto tipo que el especificado en el tercer parámetro de `CREA_FICH`, se devuelve el control al programa llamante con el valor -1 en `r29`.

2. Se descompone su nombre completo (ruta) en dos partes divididas por el último carácter separador de dicha ruta. Por ejemplo, la ruta "`/Dir1/D2/D3/D4/Fich`" quedará dividida en la ruta parcial (`RP`) "`/Dir1/D2/D3/D4`" por un lado, y el nombre del fichero (`NF`) "`Fich`" por otro. Esta división se realizará del siguiente modo:

   a) Se guardará en un registro, `r_nomfich`, la dirección de memoria en que comienza el nombre del fichero `NF` (será igual a la dirección facilitada en el segundo parámetro más la longitud de la cadena de caracteres que describe la ruta parcial).

   b) Se contará el número de caracteres que componen la ruta parcial (14 en el ejemplo) y se sumará 1 para incluir el terminador `H'00` (en total, 15).

   c) Se ajustará el número obtenido en el paso anterior para que sea un múltiplo de 4 (16 en el ejemplo). Sea `TAM` el valor obtenido.

   d) Se reservará espacio en la pila para almacenar una variable de tamaño igual a `TAM` bytes. Se guardará la dirección de la pila obtenida al reservar este espacio en un registro: `r_rp`.

   e) Se copiarán todos los caracteres de la ruta parcial sobre el espacio reservado en la pila. A continuación de los mismos se escribirá el carácter terminador `H'00`.

   Si durante estas operaciones se observa que el nombre del fichero (`NF`) contiene más de cuatro caracteres, se tratará de un error, por lo que se devolverá el control al programa llamante con el valor -1 en `r29`.

3. Si la `RP` obtenida en el paso anterior está vacía o coincide con el carácter `H'2F`, será porque se ha llegado a la raíz del disco. En este caso, el proceso continuará por el paso 4. En caso contrario, se llamará recursivamente a esta misma subrutina

(`CREA_FICH`), pasándole como parámetros la dirección del disco, la dirección guardada en `r_rp`, obtenida en el paso $2d$ y una palabra cuyo byte menos significativo tenga el valor 'D' (directorio).

4. Se procederá a crear el fichero cuyo nombre (`NF`) está almacenado en la dirección que contiene el registro `r_nomfich`. Si la ruta parcial (`RP`) obtenida en el paso 2 está vacía o coincide con el carácter `H'2F`, se creará el fichero mediante una llamada a la subrutina `CREA_RAIZ`. En caso contrario se obtendrá mediante una llamada a la subrutina `CREA_ED`.

5. Devuelve el control al programa o subrutina llamante, una vez haya copiado en `r29` la dirección obtenida como resultado de la llamada a la subrutina indicada en el paso 4 (`CREA_RAIZ` o `CREA_ED`).

Borra un archivo en un directorio especificado

```
DirEnt = BORRA_ARCH (DirDisco, DirArch)
```

Parámetros:

- **DirDisco:** Es la dirección de memoria del 88110 en que comienza el disco simulado. A partir de esa dirección se encuentran las sucesivas agrupaciones que componen el disco.
- **DirArch:** Es la dirección de memoria en la que se encuentra la entrada donde está alojado el archivo que se desea borrar.

Los dos parámetros de esta función se pasan en la pila.

Resultado:

- **DirEnt:** La rutina dejará con `H'E5` el primer byte de la entrada donde está alojado el archivo y dejará libres (con `H'00`) todas las entradas de la FAT que ocupaba el archivo. En el registro `r29` se devolverá la dirección que se pasó como parámetro `DirArch` y en caso de existir cualquier error se devolverá -1.

Descripción:

La rutina deberá comprobar que el parámetro DirArch corresponde a un archivo (no a un directorio ni a una entrada libre ni borrada). En caso contrario devolverá error. Después dejará con `H'E5` el primer byte de la entrada donde está alojado el archivo. Finalmente, recorrerá todas las entradas de la FAT ocupadas por el archivo y las dejará todas con el valor `H'00`.

Borra un directorio y todo su contenido

```
DirEnt = BORRA_DIR (DirDisco, DirDirec)
```

Parámetros:

- **DirDisco:** Es la dirección de memoria del 88110 en que comienza el disco simulado. A partir de esa dirección se encuentran las sucesivas agrupaciones que componen el disco.
- **DirDirec:** Es la dirección de memoria en la que se encuentra la entrada donde está alojado el directorio que se desea borrar.

Los dos parámetros de esta función se pasan en la pila.

Resultado:

- **DirEnt:** La rutina dejará con `H'E5` el primer byte de la entrada donde está alojado el directorio que se desea borrar y dejará libres (con `H'00`) todas las entradas de la FAT que ocupaba el directorio. Esta misma operación la realizará con todos los ficheros (archivos y directorios) que se encuentren en cualquier ruta que pueda originarse con el directorio a borrar. En el registro `r29` se devolverá la dirección que se pasó como parámetro DirDirec y en caso de existir cualquier error se devolverá -1.

Descripción:

La rutina borra de forma recursiva todos los directorios dependientes del que se quiere borrar. La rutina `BORRA_DIR` se implementará mediante un algoritmo recursivo y se desarrollará según los pasos indicados a continuación:

1. Se comprueba si el parámetro DirDirec corresponde a un directorio no borrado (no a un archivo ni a un directorio borrado, ni a una entrada libre): en caso contrario se devuelve error.
2. Se inicializa un puntero (P_Fich) con la primera entrada del primer sector que ocupa el directorio y se carga un registro (r_fat) con el valor de la primera agrupación que ocupa el directorio en la FAT.
3. Se inicializa el contador de entradas de una agrupación (Cont=8).
4. Si el primer byte de P_Fich es `H'00` ir a 8.
5. Si el primer byte de P_Fich es `H'E5` ir a 8.
6. Si P_Fich apunta a un archivo se pasan los parámetros y se llama a BORRA_ARCH. Si `r29`=-1 se devuelve error y en caso contrario ir a 8.
7. Si P_Fich apunta a un directorio se pasan los parámetros y se llama a BORRA_DIR. Previo al paso de parámetros se salvan r_fat y Cont para recuperarlos en el punto en que se dejaron.
8. Se incrementa P_Fich a la siguiente entrada y se decrementa Cont. Si Cont no es cero ir a 4. Si Cont=0 se comprueba si se ha recorrido la última agrupación del directorio:
   a) Si es la última, se carga `H'E5` en el primer byte de DirDirec, se ponen a `H'00` las entradas de la FAT que ocupa ese directorio y se retorna al llamante.
   b) Si no es la última agrupación, se accede a la siguiente agrupación del directorio y se va a 3.

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 6).

Supongamos que el registro `r30` contiene el valor 10000 (decimal) y el registro `r2` contiene el valor hexadecimal `H'04030201`. 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 6.

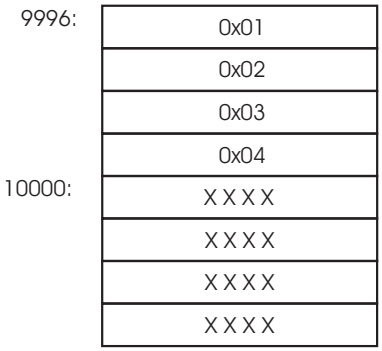

Figura 6. Operaciones PUSH y POP.

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 = `H'04030201` 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 esta práctica, 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 de la práctica se utilizará el registro `r31`. Por tanto, las primeras instrucciones de una subrutina en la que se 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 esta práctica 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.

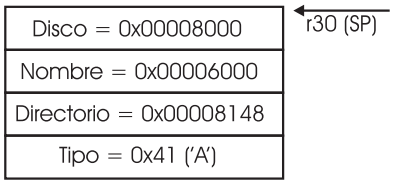

Figura 7. Paso de parámetros.

El paso de parámetros a todas las subrutinas de esta práctica se realizará utilizando la pila de la máquina y la devolución del resultado se realizará a través del registro `r29`.

El parámetro que queda en la cima de la pila cuando se hace una llamada es el primero de la lista de argumentos. Considerando el ejemplo descrito como **Caso9** en la página 27, correspondiente a la creación del archivo `CONV` en el directorio `SD22` que realiza la rutina `CREA_ED`, en el momento de invocar dicha rutina el estado de la pila sería el que se muestra en la figura 7. Dicho estado es el correspondiente al instante anterior a la llamada a la rutina, y coincide con el que se observará justo al finalizar la rutina, ya que el resultado se devuelve a través de un registro, por lo que la pila no debe verse afectada.

En el caso de la rutina `CREA_FICH`, al igual que ocurre con frecuencia a la hora de realizar otros programas en ensamblador, necesitará hacer uso de variables locales, que se ubicarán en el marco de pila. En el caso concreto de `CREA_FICH`, necesita reservar espacio para una cadena de caracteres que identifica a la ruta parcial de un fichero (veáse la descripción de la rutina `CREA_FICH`, en la página 16). Suponiendo que la ruta parcial fuera `/DIRN/S1/S2/S3`, como corresponde al ejemplo descrito como **Caso10** en la página 28, se requeriría reservar 4 palabras para la ruta parcial (un número entero de palabras, suficiente para almacenar los 15 bytes de que consta la ruta incluído el carácter terminador). Así pues,

la estructura de la pila para este caso, una vez reservadas las variables locales, sería la mostrada en la figura 8. La reserva del espacio de la variable local utilizada para almacenar la ruta parcial de dicho ejemplo, suponiendo que `r8` almacenase la longitud de dicha ruta parcial, y suponiendo que estuviese definida la macro `PUSH`, podría realizarse del siguiente modo:

```
CREA_FICH:
        PUSH  (r1)       ; Se guarda la dirección de retorno
        PUSH  (r31)      ; Se salva el FP del llamante
        addu r31,r30,r0  ; r31 <- r30 Activación puntero de marco de pila

        . . . . . .
        . . . . . .

        clr     r7, r8, 2<0>      ; anular los 2 bits menos significativos
        addu    r7, r7, 4         ; asegurar el espacio necesario
        sub     r30, r30, r7      ; reservar el espacio en pila para poner
                                  ; la parte de DIR de la RUTA
```

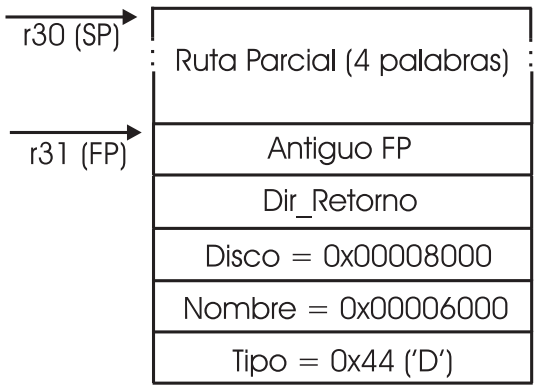

Figura 8. Gestión de variables locales.

Asignación de etiquetas y de memoria

El punto de entrada de cada una de las subrutinas deberá ir asociado a las etiquetas `BUSCA_RAIZ`, `BUSCA_ED`, `BUSCA_FICH`, `CREA_RAIZ`, `CREA_ED`, `CREA_FICH`, `BORRA_ARCH` y `BORRA_DIR` (todas las letras en mayúsculas). Por ejemplo, la primera instrucción perteneciente a la subrutina `BUSCA_FICH` deberá ir precedida de esta etiqueta:

```
BUSCA_FICH: 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** (se aconseja inicializar el puntero de pila a 0x00010000).

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. Una práctica que no se atenga a esta norma se evaluará como suspensa aún en el caso de que llege 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
```

Busca un fichero en el directorio raíz

Caso 1. Busca el fichero DOS en el directorio raíz del ejemplo del enunciado (BUSCA_RAIZ)

Situación inicial:

```
Registros:
r30=0x0000ea60 (60000)

Direcciones de memoria:
60000       00800000     00600000
24576       444F5300
```

Resultado:

```
Registros:
r29=0x00008058 (32856) r30=0x0000ea60 (60000)
```

Nota: Ninguna entrada del disco debe modificarse.

Caso 2. Busca el fichero UNIX en el directorio raíz del ejemplo del enunciado (BUSCA_RAIZ)

Situación inicial:

```
Registros:
r30=0x0000ea60 (60000)

Direcciones de memoria:
60000       00800000     00600000

24576       554E4958
```

Resultado:

```
Registros:
r29=0xFFFFFFFF (-1) r30=0x0000ea60 (60000)
```

Nota: Ninguna entrada del disco debe modificarse.

Busca un fichero en un directorio especificado

Caso 3. Busca el fichero SAPI en el directorio SD22 del ejemplo del enunciado (BUSCA_ED)

Situación inicial:

```
Registros:
r30=0x0000ea60 (60000)

Direcciones de memoria:
60000       00800000     00600000     48810000

24576       53415049
33096       53443232     440A0000
```

Resultado:

```
Registros:
r29=0x000082A8 (33448) r30=0x0000ea60 (60000)
```

Nota: Ninguna entrada del disco debe modificarse.

Caso 4. Busca el fichero TRIB en el directorio SD22 del ejemplo del enunciado (BUSCA_ED)

Situación inicial:

```
Registros:
r30=0x0000ea60 (60000)

Direcciones de memoria:
60000       00800000     00600000     48810000

24576       54524942
33096       53443232     440A0000
```

Resultado:

```
Registros:
r29=0x00008480 (33920) r30=0x0000ea60 (60000)
```

Nota: Ninguna entrada del disco debe modificarse.

Caso 5. Busca el fichero BDAT en el directorio SD22 del ejemplo del enunciado (BUSCA_ED)

Situación inicial:

```
Registros:
r30=0x0000ea60 (60000)

Direcciones de memoria:
60000       00800000     00600000     48810000

24576       42444154
```

Resultado:

```
Registros:
r29=0xFFFFFFFF (-1) r30=0x0000ea60 (60000)
```

Nota: Ninguna entrada del disco debe modificarse.

Busca un fichero a partir de su nombre completo

Caso 6. Busca el fichero /DIR2/SD22/SSTR/Z en el disco del ejemplo del enunciado (BUSCA_FICH)

Situación inicial:

```
Registros:
r30=0x0000ea60 (60000)

Direcciones de memoria:
60000       00800000     00600000

24576       2F444952     322F5344     32322F53     5354522F
24592       5A000000
```

Resultado:

```
Registros:
r29=0x00008990 (35216) r30=0x0000ea60 (60000)
```

Nota: Ninguna entrada del disco debe modificarse.

Caso 7. Busca el fichero /DIR2/SD22/SSTR/ACTA en el disco del ejemplo del enunciado (BUSCA_FICH)

Situación inicial:

```
Registros:
r30=0x0000ea60 (60000)

Direcciones de memoria:
60000       00800000     00600000

24576       2F444952     322F5344     32322F53     5354522F
24592       41435441     00000000
```

Resultado:

```
Registros:
r29=0x00008998 (35224) r30=0x0000ea60 (60000)
```

Nota: Ninguna entrada del disco debe modificarse.

Crea un fichero en el directorio raíz

Caso 8. Crea el fichero PRU de tipo "archivo" en el directorio raíz del ejemplo del enunciado (CREA_RAIZ)

Situación inicial:

```
Registros:
r30=0x0000ea60 (60000)

Direcciones de memoria:
60000       00800000     00600000     41000000

24576       50525500

32832       44495231     44040000     44495232     44050000
32848       494F0000     41060064     444F5300     41080032
32864       E5495233     44090000     54445000     44200000
32880       00000000     00000000     00000000     00000000

32960       FFFFFFFF     FFFF07FF     FFFF1200     00FF14FF
32976       FFFFFF00     1D00FFFF     FFFF0000     0025002C
32992       FF000000     002AFFFF     3000FF00     2D310000
33008       FFFF33FF     FFFF38FF     FF000000     00000000
```

Resultado:

```
Registros:
r29=0x00008060 (32864) r30=0x0000ea60 (60000)
```

```
Direcciones de memoria:
32832       44495231     44040000     44495232     44050000
32848       494F0000     41060064     444F5300     41080032
32864       50525500     410B0000     54445000     44200000
32880       00000000     00000000     00000000     00000000

32960       FFFFFFFF     FFFF07FF     FFFF12FF     00FF14FF
32976       FFFFFF00     1D00FFFF     FFFF0000     0025002C
32992       FF000000     002AFFFF     3000FF00     2D310000
33008       FFFF33FF     FFFF38FF     FF000000     00000000
```

Nota: Ninguna otra entrada del disco debe modificarse.

Crea un fichero en un directorio especificado

Caso 9. Crea el fichero CONV de tipo "archivo" en el directorio SD22 del ejemplo del enunciado (CREA_ED)

Situación inicial:

```
Registros:
r30=0x0000ea60 (60000)

Direcciones de memoria:
60000       00800000     00600000     48810000     41000000

24576       434F4E56

33408       434F4F52     410E012C     534C4142     440F0000
33424       53455354     44100000     53415351     44110000
33440       53544643     44160000     53415049     44170000
33456       53444D43     44180000     53444543     44190000

33920       54524942     411F00FA     E5495233     44090000
33936       53535452     44260000     53414D50     44270000
33952       00000000     00000000     00000000     00000000
33968       00000000     00000000     00000000     00000000

32960       FFFFFFFF     FFFF07FF     FFFF1200     00FF14FF
32976       FFFFFF00     1D00FFFF     FFFF0000     0025002C
32992       FF000000     002AFFFF     3000FF00     2D310000
33008       FFFF33FF     FFFF38FF     FF000000     00000000
```

Resultado:

```
Registros:
r29=0x00008488 (33928) r30=0x0000ea60 (60000)
```

```
Direcciones de memoria:
33920       54524942     411F00FA     434F4E56     410B0000
33936       53535452     44260000     53414D50     44270000
33952       00000000     00000000     00000000     00000000
33968       00000000     00000000     00000000     00000000

32960       FFFFFFFF     FFFF07FF     FFFF12FF     00FF14FF
32976       FFFFFF00     1D00FFFF     FFFF0000     0025002C
32992       FF000000     002AFFFF     3000FF00     2D310000
33008       FFFF33FF     FFFF38FF     FF000000     00000000
```

Nota: Ninguna otra entrada del disco debe modificarse.

Crea un fichero cuyo nombre completo se especifica

Caso 10. Crea el fichero /DIRN/S1/S2/S3/D de tipo "directorio" en el disco del ejemplo del enunciado (CREA_FICH)

Situación inicial:

```
Registros:
r30=0x0000ea60 (60000)

Direcciones de memoria:
60000       00800000     00600000     44000000

24576       2F444952     4E2F5331     2F53322F     53332F44
24592       00000000

32832       44495231     44040000     44495232     44050000
32848       494F0000     41060064     444F5300     41080032
32864       E5495233     44090000     54445000     44200000
32880       00000000     00000000     00000000     00000000

32960       FFFFFFFF     FFFF07FF     FFFF1200     00FF14FF
32976       FFFFFF00     1D00FFFF     FFFF0000     0025002C
32992       FF000000     002AFFFF     3000FF00     2D310000
33008       FFFF33FF     FFFF38FF     FF000000     00000000
```

Resultado:

```
Registros:
r29=0x00008540 (34112) r30=0x0000ea60 (60000)

Direcciones de memoria:
32832       44495231     44040000     44495232     44050000
32848       494F0000     41060064     444F5300     41080032
32864       4449524E     440B0000     54445000     44200000
32880       00000000     00000000     00000000     00000000
```

```
33472       53310000     440C0000
33536       53320000     44130000
33984       53330000     44150000
34112       44000000     441A0000

32960       FFFFFFFF     FFFF07FF     FFFF12FF     FFFF14FF
32976       FFFFFFFF     1DFFFFFF     FFFFFF00     0025002C
32992       FF000000     002AFFFF     3000FF00     2D310000
33008       FFFF33FF     FFFF38FF     FF000000     00000000
```

Nota: Ninguna otra entrada del disco debe modificarse.

Caso 11. Crea el fichero /DIR2/SD22/SSTR/INCI de tipo "archivo" en el disco del ejemplo del enunciado (CREA_FICH)

Situación inicial:

```
Registros:
r30=0x0000ea60 (60000)

Direcciones de memoria:
60000       00800000     00600000     41000000

24576       2F444952     322F5344     32322F53     5354522F
24592       494E4349     00000000

35200       5041434F     41340032     E5495233     44290000
35216       5A000000     4136006A     41435441     44370000
35232       00000000     00000000     00000000     00000000
35248       00000000     00000000     00000000     00000000

32960       FFFFFFFF     FFFF07FF     FFFF1200     00FF14FF
32976       FFFFFF00     1D00FFFF     FFFF0000     0025002C
32992       FF000000     002AFFFF     3000FF00     2D310000
33008       FFFF33FF     FFFF38FF     FF000000     00000000
```

Resultado:

```
Registros:
r29=0x00008988 (35208) r30=0x0000ea60 (60000)

Direcciones de memoria:
35200       5041434F     41340032     494E4349     410B0000
35216       5A000000     4136006A     41435441     44370000
35232       00000000     00000000     00000000     00000000
35248       00000000     00000000     00000000     00000000
```

```
32960       FFFFFFFF     FFFF07FF     FFFF12FF     00FF14FF
32976       FFFFFF00     1D00FFFF     FFFF0000     0025002C
32992       FF000000     002AFFFF     3000FF00     2D310000
33008       FFFF33FF     FFFF38FF     FF000000     00000000
```

Nota: Ninguna otra entrada del disco debe modificarse.

Borra un archivo en un directorio especificado

Caso 12. Borra el archivo COOR del ejemplo del enunciado (BORRA_ARCH)

Situación inicial:

```
Registros:
r30=60000 (0x0000ea60)

Direcciones de memoria:
60000       00800000     80820000

33408       434F4F52     410E012C

32960       FFFFFFFF     FFFF07FF     FFFF1200     00FF14FF
32976       FFFFFF00     1D00FFFF     FFFF0000     0025002C
32992       FF000000     002AFFFF     3000FF00     2D310000
33008       FFFF33FF     FFFF38FF     FF000000     00000000
```

Resultado:

```
Registros:
r29=0x00008280 r30=60000 (0x0000ea60)

Direcciones de memoria:
33408       E54F4F52     410E012C

32960       FFFFFFFF     FFFF07FF     FFFF1200     00FF00FF
32976       FFFFFF00     0000FFFF     FFFF0000     0000002C
32992       FF000000     0000FFFF     30000000     2D310000
33008       FFFF33FF     FFFF38FF     FF000000     00000000
```

Nota: Ninguna otra entrada del disco debe modificarse.

Borra un directorio y todo su contenido

Caso 13. Borra el directorio SD22 del ejemplo del enunciado. (BORRA_DIR)
Situación inicial:

```
Registros:
r30=60000 (0x0000ea60)
Direcciones de memoria:
60000       00800000     48810000
```

```
33088                                   53443232    440A0000

33408       434F4F52    410E012C    534C4142    440F0000
33424       53455354    44100000    53415351    44110000
33440       53544643    44160000    53415049    44170000
33456       53444D43    44180000    53444543    44190000

33728       42444154    41280078    50524F42    41320055
33744       41435441

33920       54524942    411F00FA    E5495233    44090000
33936       53535452    44260000    53414D50    44270000

35200       5041434F    41340032    E5495233    44290000
35216       5A000000    4136006A    41435441    44370000

32960       FFFFFFFF    FFFF07FF    FFFF1200    00FF14FF
32976       FFFFFF00    1D00FFFF    FFFF0000    0025002C
32992       FF000000    002AFFFF    3000FF00    2D310000
33008       FFFF33FF    FFFF38FF    FF000000    00000000
```

Resultado:

```
Registros:
r29=0x00008148 r30=60000 (0x0000ea60)

Direcciones de memoria:
33088                                   E5443232    440A0000

33408       E54F4F52    410E012C    E54C4142    440F0000
33424       E5455354    44100000    E5415351    44110000
33440       E5544643    44160000    E5415049    44170000
33456       E5444D43    44180000    E5444543    44190000

33728       E5444154    41280078    E5524F42    41320055
33744       E5435441    44350000

33920       E5524942    411F00FA    E5495233    44090000
33936       E5535452    44260000    E5414D50    44270000

35200       E541434F    41340032    E5495233    44290000
35216       E5000000    4136006A    E5435441    44370000

32960       FFFFFFFF    FFFF07FF    FFFF0000    00FF0000
32976       00000000    00000000    00000000    00000000
32992       FF000000    00000000    00000000    00000000
33008       00000000    00000000    00000000    00000000
```

Nota: Ninguna otra entrada del disco debe modificarse.

NORMAS DE PRESENTACIÓN

Toda la información relativa a esta práctica, incluyendo las fechas de entrega/corrección/examen para cada convocatoria, se dejará disponible en la dirección:

`http://www.datsi.fi.upm.es/docencia/Lab_Estructura/Ensamblador`

Dicha página contiene una sección de anuncios relacionados con la práctica.

TUTORÍAS DE LA PRÁCTICA

Las posibles preguntas relacionadas con la práctica se atenderán por correo electrónico en la dirección (**pr_ensamblador@datsi.fi.upm.es**) o personalmente en los despachos 4105 y/o 4106. El horario de atención personal a los alumnos para cuestiones relacionadas con esta práctica es el que se especifica para los profesores encargados durante el presente curso académico (Manuel M. Nieto y José L. Pedraza) en la siguiente dirección:

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

ENTREGA DE LA PRÁCTICA

La entrega se compone de:

1. Una **memoria**, en formato DINA4, en cuya portada deberá figurar claramente el nombre y apellidos del **autor** de la práctica, su identificador en el sistema de entregas y el nombre de la asignatura.

   La memoria se entregará en formato (PDF) mediante un mensaje enviado a la dirección (**pr_ensamblador@datsi.fi.upm.es**) y especificando en el "Asunto": "Memoria de práctica LEC" seguido del identificador del grupo de prácticas.

   La memoria deberá contener los siguientes puntos:

   - Descripción de los algoritmos que se han implementado para cada subrutina, especificando la utilización de la pila que se hace en la misma.

     Se podrá realizar esta descripción como considere más conveniente: mediante un diagrama de flujo, utilizando pseudocódigo, con una descripción textual, o mediante cualquier lenguaje que facilite la comprensión de la labor realizada por la subrutina para implementar lo especificado en el enunciado. La descripción no se debe limitar a repetir la especificación, sino que debe incidir en las particularidades de la implementación, por lo que se recomienda elaborar la memoria al mismo tiempo que se desarrollan los programas en ensamblador.
   - Descripción del juego de ensayo (conjunto de casos de prueba) que el alumno haya diseñado y utilizado para probar el correcto funcionamiento de la práctica. Se especificarán las pruebas realizadas para probar cada una de las subrutinas, señalando los casos particulares que se han probado e incluyendo el código de los más representativos.
   - Observaciones finales y comentarios personales de esta práctica, entre los que se debe incluir una estimación del tiempo empleado en su realización y el porcentaje aproximado de implicación en la práctica de cada uno de los miembros del grupo.

NOTA: No será necesario incluir en esta memoria una copia del listado de las subrutinas en ensamblador, aunque sí será necesario que dichas rutinas se encuentren adecuadamente comentadas en el propio fichero que se entrega para las correcciones, ya que dicho fichero será consultado en el proceso de evaluación de la práctica.

2. La entrega de los ficheros que contienen la práctica. 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 de la práctica. La práctica se realizará individualmente o en grupos de **dos alumnos**. Cada línea de este fichero contendrá los datos de uno de los autores de la práctica, 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 también se admiten direcciones correspondientes a otros proveedores de correo electrónico.
   - **dosfat.ens**: Contendrá las subrutinas que componen la práctica junto con un programa principal que se haya utilizado para su depuración.
   - **memoria.txt**: Será un fichero ASCII que contenga la memoria de la práctica. Si realiza esta memoria mediante un procesador de textos, será suficiente con que guarde una copia del fichero en formato ASCII (.txt), e incluya dicho fichero.
     Si la versión impresa de la memoria incluye diagramas de flujo u otros elementos gráficos, no es necesario que incorpore estos componentes en el fichero **memoria.txt**.

IMPORTANTE: Se recomienda que antes de realizar una entrega de la práctica, realicen el ensamblado del fichero **dosfat.ens**, se aseguren de que no genera ningún error y ejecuten la práctica con sus propios casos de prueba. De este modo reducirán 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 de prácticas. 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 de prácticas 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 mandar los ficheros de una práctica, que deberán estar en el directorio de trabajo del usuario. Si el comando se ejecuta correctamente se mostrarán los siguientes mensajes:

MANDANDO EL FICHERO dosfat.ens ...OK.

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

MANDANDO EL FICHERO dosfat.ens ...
No se puede abrir el fichero dosfat.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 dosfat.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 grupo de prácticas será eliminado de la lista de prácticas pendientes de corregir. Si se han realizado varias entregas se cancelarán todas las entregas.

CONSULTAR RESULTADOS

Esta opción permite consultar los resultados de la corrección de la entrega de una práctica. 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 las prácticas. El alumno no debe utilizar este programa para depurar su práctica. Debe ser el propio alumno el que construya su conjunto de pruebas que le permita comprobar que la práctica funciona correctamente.

BLOQUEO DE LA ENTREGA

Si el usuario se compromete a no entregar más veces la práctica, 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 a la práctica. 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 de la práctica.

NOTICIAS

Esta opción es puramente informativa. Permite notificar al alumno modificaciones en la especificación de la práctica o, en general, noticias de interés de la asignatura asociada a la práctica. 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 práctica 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 la práctica 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 de la práctica y teclear lo siguiente (suponiendo que los ficheros se encuentran en una carpeta denominada "PRACTICA" dentro de un pen-drive o lápiz de datos que se ubica en la unidad "H:" de Windows):

```
G:\>h:
H:\>cd practica
H:\PRACTICA>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 de la práctica deberán tomar las debidas precauciones para evitar la difusión o copia de su código. ¡Se recomienda trabajar siempre sobre el lápiz de datos 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 pueden extraer de las normas de la asignatura.

Sistema de entrega vía Web

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 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 desde dicho computador.

A.0.1. Obtención del entorno de prácticas

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

`http://www.datsi.fi.upm.es/docencia/Lab_Estructura/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 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. 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 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:

  $C:\backslash Facultad\backslash Ensamblador$
- 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:

  $cd\ \ C:\backslash Facultad\backslash Ensamblador\backslash 88k\_Windows$

  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. Puede utilizar por ejemplo el editor "Wordpad", pero no debe emplear procesadores de texto como "Word" u otros.