

Arquitectura de Computadores

Práctica de Memorias Cache

Curso 2011/2012

Departamento de Arquitectura y Tecnología de Sistemas Informáticos
Universidad Politécnica de Madrid

Octubre de 2011.

1. INTRODUCCIÓN

Esta práctica trata de profundizar en el estudio de las memorias cache mediante la experimentación de su funcionamiento sobre un conjunto de programas sencillos. En todos los casos, se analizará el comportamiento de la memoria cache desde diferentes perspectivas:

1. Observando la influencia de sus distintos parámetros de diseño (tamaño total, tamaño de bloque y políticas de ubicación y escritura) en la ejecución del programa.
2. Identificando las distintas causas que hacen que se produzcan fallos en la memoria cache.
3. Comprobando cómo pequeñas modificaciones en el código de los programas pueden influir en las prestaciones obtenidas.

Nos centraremos exclusivamente en el comportamiento de la **cache de datos** de una máquina, el Motorola MC88110, que dispone de caches separadas para datos e instrucciones. Para ello se empleará el emulador de este procesador utilizado ya en la asignatura *Laboratorio de Estructura de Computadores*. Así, los programas que se utilizan aquí están escritos en un lenguaje ensamblador y para una arquitectura ya conocidos por el alumno. Para la realización de esta práctica se ha modificado la configuración del emulador, al que se le dota ahora de memorias cache. Es ésta la razón por la que cambia la forma en que éste se invoca con respecto a la que se utilizó en aquella asignatura.

Adicionalmente, se proporciona al alumno una nueva herramienta que le permitirá configurar la cache de datos. Esta herramienta permite seleccionar los parámetros de diseño de la cache antes mencionados, pero no la **política de lectura**, que está predefinida como *out of order fetch*, y la de **reemplazo**, que está predefinida como *LRU*.

Por otra parte, debe tenerse en cuenta que el comportamiento del sistema de memoria ante un **fallo de escritura** en la cache de datos cuando ésta se ha configurado como *copy-back* es el siguiente: en primer lugar se realiza la escritura de la palabra que ha producido el fallo sobre la memoria principal y, a continuación, se transfiere el bloque de memoria principal que contiene dicha palabra a la memoria cache.

La práctica se realizará en los terminales del Centro de Cálculo (**Sala Reguerillo**, conectándose al computador **batman**). Los alumnos que dispongan de un computador con sistema operativo Linux, disponen de una versión del emulador para dicho sistema (véanse las notas de instalación en la hoja de Web de la práctica).

El resto de este documento se estructura de la siguiente forma:

- **DEFINICIONES:** se presentan aquí algunas definiciones, fundamentalmente relativas a los tipos de fallos de cache, que se manejarán en los ejercicios a desarrollar.
- **TÉCNICAS DE MEJORA:** en este apartado se describen someramente diferentes modificaciones del código que pueden afectar al rendimiento, siempre desde el punto de vista del comportamiento de la memoria cache.
- **EMULADOR:** aquí se describe el uso del emulador, y en particular los mandatos de depuración.

- **TUTORÍAS:** horarios de tutoría sobre la práctica de los profesores encargados de la misma.
- **NORMAS:** por último, se incluyen las normas de entrega de la práctica: procedimiento y plazos.

2. DEFINICIONES

Se utilizará en esta práctica la clasificación de las posibles causas de los fallos de cache ya estudiada en las clases teóricas (fallos forzosos, por capacidad y por conflicto).

Para la identificación de los **fallos forzosos** se ha de seguir siempre de forma estricta la definición de dicho tipo de fallos (los que se producen cuando se hace referencia por primera vez a un bloque que no está en la cache).

Por otro lado, es a veces muy difícil distinguir, a nivel individual, si un fallo es de conflicto o es de capacidad. Para facilitar esta labor, se proporciona a continuación una definición ampliada de estos dos tipos de fallos y un procedimiento práctico para realizar su clasificación:

- **Fallos por capacidad:** Se deben a que la memoria cache no puede contener todos los bloques que se necesitan durante la ejecución de un programa. Se producen cuando se accede a un bloque de memoria principal que ha estado previamente en la memoria cache, pero que ha dejado de estarlo, puesto que tuvo que ser reemplazado al llenarse la memoria cache.
- **Fallos por conflicto:** Se producen **únicamente** en memorias cache con organización directa o asociativa por conjuntos. Se deben a que, en estos casos, hay múltiples bloques de memoria principal que se asignan al mismo bloque o conjunto de bloques de memoria cache.

A la vista de las definiciones anteriores, el **método práctico** más adecuado para **diferenciar** los fallos por **capacidad** de los fallos por **conflicto** consistirá en lo siguiente:

1. Determinar el número de fallos por capacidad. Para ello, se partirá de una memoria cache totalmente asociativa y con **política de reemplazo óptima**. Sabiendo el número total de fallos y el número de fallos forzosos durante la ejecución de un cierto programa, se obtendrá el número de fallos por capacidad sin más que restar al número total de fallos obtenido, el número de ellos que se producen por la carga inicial (forzosos).
2. Generalizar para otras configuraciones. Otras memorias cache del mismo tamaño y características generales salvo su organización (i.e., política de ubicación), producirán el mismo número de fallos por capacidad al ejecutar un programa con la misma traza que el empleado en el paso 1.
3. Obtener el número de fallos por conflicto. Conociendo, para una configuración particular, el número total de fallos, el número de fallos de carga inicial y el número de fallos por capacidad, el cálculo del número de fallos por conflicto se realizará sin más que restar del total, la suma de los fallos de carga inicial más los de capacidad.

3. TÉCNICAS DE MEJORA

Se describen aquí, de manera genérica, diferentes técnicas que pueden ayudar a mejorar el tiempo de ejecución de un programa mediante la modificación de su código. En todos los casos se trata de reducir el número de fallos de cache producidos, aunque en varios de ellos se influye también en el número de instrucciones ejecutadas. Estas técnicas son las siguientes: **intercambio de bucles** (*loop interchanging*), **fusión de bucles** (*loop fusion*), **fisión o distribución de bucles** (*loop fission*), **desenrollado de bucles** (*loop unrolling*), *strip mining*, **reemplazo escalar**, *merging arrays* y **operación por bloques**.

3.1. Intercambio de bucles

Consiste en intercambiar los bucles que se utilizan para recorrer una matriz: por filas o por columnas. El objetivo es que la matriz se recorra en el mismo orden en que está almacenada en memoria, lo que aumentará la proximidad de referencias espacial del programa.

Ejemplo:

```
/* Matriz almacenada por filas */          /* Se recorre por filas */
for (j=0; j<32; j++)                      for (i=0; i<32; i++)
  for (i=0; i<32; i++)                    ==>   for (j=0; j<32; j++)
    mat[i][j] = (i+j)%32                  mat[i][j] = (i+j)%32
```

3.2. Fusión de bucles

Consiste en unir en uno solo los diferentes bucles sucesivos en los que se utiliza la misma información (una matriz, vector, etc.).

Ejemplo:

```
for (i=0; i<32; i++)                      for (i=0; i<32; i++)
  for (j=0; j<32; j++)                    for (j=0; j<32; j++)
    mat[i][j] = (i+j) % 32;                {
                                          mat[i][j] = (i+j)%32;
                                          suma[i] = suma[i] + mat[i][j];
                                          cuenta[j] = cuenta[j] + mat[i][j];
                                          }
                                          ==>
for (i=0; i<32; i++)
  for (j = 0; j<32; j++)
    suma[i] = suma[i] + mat[i][j];

for (i=0; i<32; i++)
  for (j = 0; j<32; j++)
    cuenta[j] = cuenta[j] + mat[i][j];
```

3.3. Fisión o distribución de bucles

Consiste en transformar un bucle en varios bucles, de modo que en cada uno de ellos se trate sólo una parte del cuerpo del bucle original. Realiza la operación inversa a la fusión de bucles.

Ejemplo:

```
for (i=0; i<32; i++)
  for (j=0; j<32; j++)
    { mat[i][j] = (i+j)%32;
      suma[i][j] = 0;
    }
==>
for (i=0; i<32; i++)
  for (j=0; j<32; j++)
    mat[i][j] = (i+j)%32;
for (i=0; i<32; i++)
  for (j=0; j<32; j++)
    suma[i][j] = 0;
```

3.4. Desenrollado de bucles y *Unroll and jam*

La idea básica de esta técnica, así como de la que se explica en el siguiente apartado, es reutilizar los datos presentes en la memoria cache. Consiste en replicar el bucle más interno un número determinado de veces.

Ejemplo:

```
for (i=0; i<2*M; i++)
  for (j=0; j<N; j++)
    a[i] = a[i] + b[j];
==>
for (i=0; i<2*M; i+=2)
  { for (j=0; j<N; j++)
    a[i] = a[i] + b[j];
    for (j=0; j<N; j++)
    a[i+1] = a[i+1] + b[j];
  }
```

Aplicando posteriormente **fusión** de bucles, se obtendría la optimización denominada *unroll and jam*, cuyo efecto se muestra a continuación:

```
for (i=0; i<2*M; i+=2)
  for (j=0; j<N; j++)
    { a[i] = a[i] + b[j];
      a[i+1] = a[i+1] + b[j];
    }
```

3.5. *Strip Mining*

Consiste en agrupar un número determinado de iteraciones para formar un nuevo bucle más interno.

Ejemplo:

```
for (i=0; i<2*M; i+=4)
  for (j=0; j<N; j++)
    { a[i] = a[i] + b[j];
      a[i+1] = a[i+1] + b[j];
      a[i+2] = a[i+2] + b[j];
      a[i+3] = a[i+3] + b[j];
    }
==>
for (i=0; i<2*M; i+=4)
  for (j=0; j<N; j++)
    for (k=i; k<i+4; k++)
      a[k] = a[k] + b[j];
```

3.6. Reemplazo escalar

La idea básica del reemplazo escalar es utilizar registros para cargar el contenido de direcciones de memoria que van a volver a ser referenciadas, sustituyendo así accesos a memoria por accesos a registros.

Explicaremos esta técnica siguiendo el ejemplo del producto de matrices. En el caso del producto de matrices cuadradas éste se puede realizar mediante tres bucles anidados:

```
for (i=0; i<N; i++)
  for (j=0; j<N; j++)
    for (k=0; k<N; k++)
      x[i][j] = x[i][j] + y[i][k] * z[k][j];
```

Este algoritmo básico realiza en cada iteración del bucle (es decir, $N*N*N$ veces) tres lecturas (x , y , z) y una escritura (x).

El reemplazo escalar consiste aquí simplemente en evitar que se realice una escritura en memoria en cada paso del algoritmo de multiplicación. Para ello, la acumulación del bucle interno se realiza sobre una variable escalar (asignada a un registro), previamente puesta a cero, con lo que se reduce notablemente el número de accesos tanto de lectura como de escritura. El programa resultante es el siguiente:

```
for (i=0; i<N; i++)
  for (j=0; j<N; j++)
  {
    r = 0;
    for (k=0; k<N; k++)
      r + = y[i][k] * z[k][j];
    x[i][j] = r;
  }
```

3.7. *Merging arrays*

Se puede utilizar cuando en el programa se hace referencia a varios arrays de la misma dimensión, utilizando los mismos índices. El problema que se puede presentar en este caso es que estos accesos interfieran entre sí, dando lugar a fallos por conflicto. La técnica de *merging arrays* consiste en combinar estos arrays independientes en un array compuesto.

Ejemplo:

```
int v1[TAM];
int v2[TAM];

for (i=0; i<TAM; i++)
  { v1[i] = 0;
    v2[i] = 0;
  }

struct mezcla { int v1;
                int v2;};

struct mezcla vector_compuesto[TAM];

for (i=0; i<TAM; i++)
  { vector_compuesto[i].v1 = 0;
    vector_compuesto[i].v2 = 0;
  }
```

También se puede utilizar para agrupar variables relacionadas, con lo que se potenciaría la proximidad espacial.

3.8. Operación por bloques (*blocking*)

La operación por bloques consiste en dividir las matrices (en general las estructuras de datos) con las que se opera en bloques de tamaño suficientemente pequeño como para que quepan en la memoria cache, independientemente del tamaño original de dichas matrices. Con ello se pretende aumentar la reutilización de la información, ya que se persigue que una vez que se ha llevado un cierto dato a memoria cache, sólo salga de ella cuando no se vaya a utilizar más.

Como ejemplo de implementación, se considerará de nuevo la multiplicación de matrices. La solución que se presenta a continuación consiste en mantener B columnas de la matriz z en la cache mientras se generan las mismas B columnas de la matriz resultado x . En la figura 1 se muestra gráficamente esta solución para matrices de 6 por 6 elementos manteniendo 3 columnas.

```
for (i=0; i<N; i++)
  for (j=0; j<N; j++)
    x[i][j] = 0;

for (j=0; j<N; j+=B)
  for (i=0; i<N; i++)
    for (k=0; k<N; k++)
      for (m=j; m<j+B; m++)
        x[i][m] = x[i][m] + y[i][k] * z[k][m];
```

Debe observarse que, puesto que cada elemento de la matriz resultado se va rellenando progresivamente a lo largo de varias pasadas, es necesario que antes de comenzar la ejecución del algoritmo, el elemento correspondiente de la matriz resultado esté inicializado a 0.

Obsérvese además que se puede aplicar reemplazo escalar al término $y[i][k]$, con lo que disminuiría el número de accesos a memoria y el programa resultante sería el siguiente:

```
for (i=0; i<N; i++)
  for (j=0; j<N; j++)
    x[i][j] = 0;

for (j=0; j<N; j+=B)
  for (i=0; i<N; i++)
    for (k=0; k<N; k++)
      { aux = y[i][k];
        for (m=j; m<j+B; m++)
          x[i][m] = x[i][m] + aux * z[k][m];
      }
```

Cuando las matrices son tan grandes que la cache no puede dar cabida a B columnas enteras, existe otra solución algo más compleja: operar sobre submatrices en lugar de operar sobre columnas completas. El programa resultante para la multiplicación de matrices sería el que se muestra a continuación. En la figura 2 se muestra gráficamente esta solución para matrices de 6 x 6 elementos manteniendo submatrices de 3 x 3 elementos.

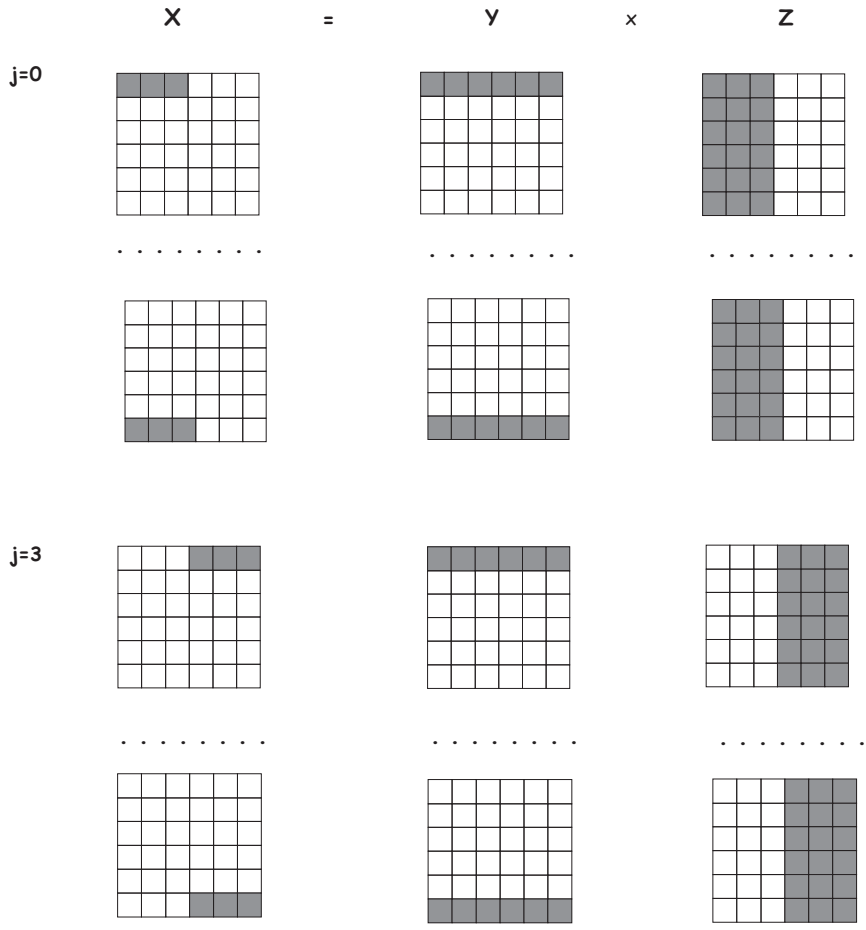


Figura 1: Operación por bloques, manteniendo en cache 3 columnas.

```

for (i=0; i<N; i++)
  for (j=0; j<N; j++)
    x[i][j] = 0;

for (j=0; j<N; j+=B)
  for (k=0; k<N; k+=B)
    for (i=0; i<N; i++)
      for (m=j; m<j+B; m++)
        { r = 0;
          for (n=k; n<k+B; n++)
            r += y[i][n] * z[n][m];
          x[i][m] += r;
        }

```

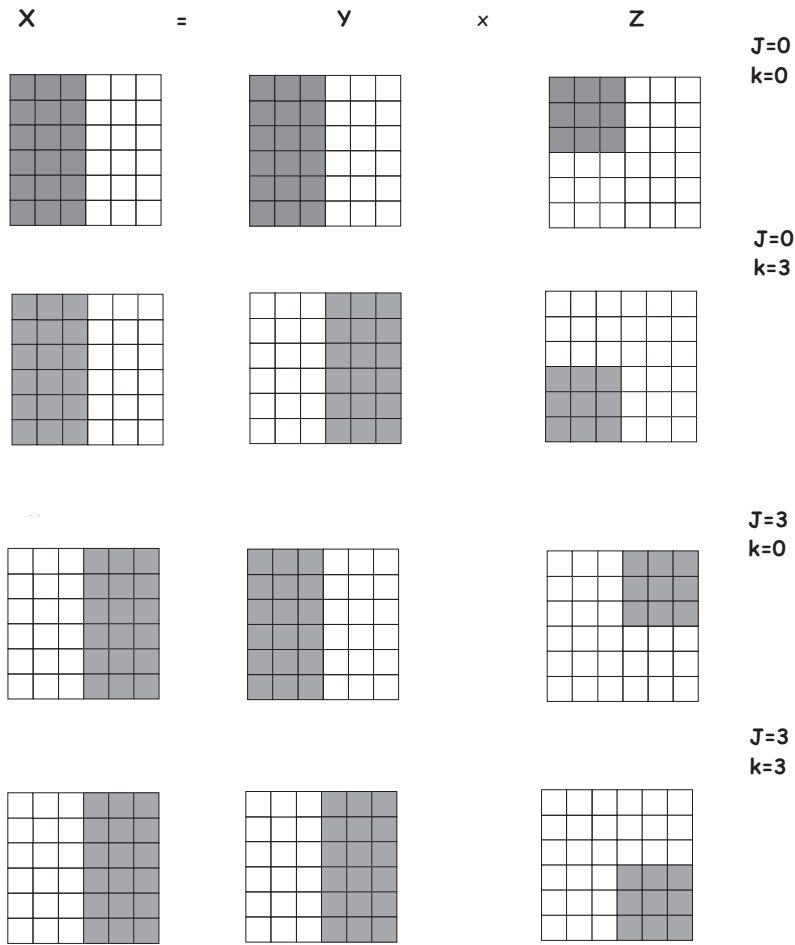


Figura 2: Operación por bloques, manteniendo en cache una submatriz de 3x3 elementos.

4. EMULADOR

El emulador del procesador MC88100 es el mismo utilizado en la asignatura *Laboratorio de Estructura de Computadores*.

4.1. Invocación del emulador

- El mandato para ejecutar el emulador es:

```
em88110 -c <fichero de configuración> <programa.bin>
```

- La memoria cache de datos se configura mediante la orden:

```
88110ins <fichero de configuración>
```

Esta orden permite configurar también la memoria principal, la memoria cache de instrucciones y las características básicas del procesador. Al generar una nueva configuración se deben definir para todos estos parámetros los valores indicados en la hoja de información general del cuestionario (Memoria principal: 10 ciclos, no entrelazada; Memoria Cache de Instrucciones: habilitada, 2 ciclos, 128 bloques de 16 bytes, directa; Registro de Control: redondeo al más próximo, ejecución secuencial, no se permiten excepciones, modo de almacenamiento *little endian*).

- Para ensamblar los programas a partir del punto de entrada solicitado:

```
88110e -e PPAL -o <fichero de salida> <fichero fuente>
```

4.2. Mandatos del emulador

La interfaz presenta un indicador en el que el usuario va introduciendo mandatos para avanzar en la ejecución del programa y ver el estado interno de la máquina. Es posible incluir puntos de ruptura para detener la ejecución. Los mandatos pueden ser diferentes cuando estamos en modo de ejecución serie o en modo de ejecución superescalar. Son los siguientes:

- H.- Presenta una ayuda. Lista todos los mandatos.
- Q.- Termina la simulación y vuelve al sistema operativo.
- P [+|- <esp_direccion>].- Si el mandato es p sin más, lista todos los puntos de ruptura activos. Si lleva un parámetro precedido del signo +, por ejemplo p + 24, incluye un punto de ruptura en la dirección 24 (expresada en decimal). Si es p - 0x14, desactiva el punto de ruptura de la dirección 14 (expresada en hexadecimal). El parámetro `esp_direccion` puede especificar una dirección (expresada en decimal o hexadecimal) o una etiqueta. Por ejemplo, si la etiqueta BUCLE está asociada a la dirección 0x1000, el comando p + BUCLE es equivalente a p + 0x1000.

- D `<esp_direccion> [<rango>]`.- Presenta un grupo de instrucciones en ensamblador a partir de la dirección especificada. El parámetro `esp_direccion` puede especificar una dirección de memoria o una etiqueta. Si se especifica el parámetro opcional `rango` (en decimal o hexadecimal) se desensamblan tantas instrucciones como se especifique en dicho parámetro. Por ejemplo el mandato `d 0x100 32` desensambla 32 instrucciones desde la dirección 100 (expresada en hexadecimal). Si no se especifica el parámetro `rango` se desensamblarán 15 instrucciones.
- V [`C`] `<esp_direccion> [<rango>]`.- El argumento `C` es opcional. Si no está presente, se visualiza el contenido de tantas posiciones de memoria principal como indique el parámetro `rango` a partir de la dirección indicada.

El parámetro `esp_direccion` puede especificar una dirección de memoria o una etiqueta. Los datos se muestran en hexadecimal y en palabras de 32 bits. Al igual que en el mandato D tanto la dirección como el `rango` pueden ir expresados en decimal o hexadecimal. El parámetro `rango` es opcional y si no se especifica se muestran 32 palabras de memoria.

Si se pasa el parámetro opcional `C`, el comando muestra posiciones de la cache de datos (si está activada). Si el contenido de una palabra de la cache es un conjunto de espacios en blanco indicará que dicha palabra no está presente en la memoria cache.

- E.- Ejecuta el programa a partir de la dirección actual del contador de programa. La ejecución no se detendrá hasta que se encuentre un punto de ruptura o la instrucción `stop`.
- T [`<veces>`].- Este mandato tiene significado distinto en los modos de ejecución *serie* y *superescalar*. En ambos modos si se omite `veces` se supone uno. En modo serie ejecutará instrucciones completas; tantas instrucciones como se indique en `veces` o una instrucción.
- R [`<numero> <valor>`].- Presenta el contenido de los registros y algunas estadísticas de acceso a las caches de datos e instrucciones (si están activadas). Si este mandato va seguido de un identificador de registro y de un valor (decimal o hexadecimal), se modificará dicho registro con el valor especificado. Por ejemplo el mandato `r 12 0x4a` carga el valor hexadecimal 4a en el registro 12 del banco.
- I `<esp_direccion> <valor>`.- Modifica el contenido de una palabra de memoria principal `esp_direccion` (en decimal o hexadecimal) con el valor que le sigue `valor` (en decimal o hexadecimal). El parámetro `esp_direccion` puede especificar una dirección de memoria o una etiqueta.
- S.- Este mandato presenta las estadísticas de acceso a las caches de instrucciones y datos (si están activadas).
- C.- Este mandato presenta los parámetros de configuración del computador emulado.

Para facilitar el uso de la interfaz de usuario del emulador, se ha incorporado un histórico de mandatos que permite acceder a los mandatos tecleados anteriormente, editarlos y ejecutarlos de nuevo mediante las teclas de movimiento de cursor. Las particularidades de manejo que incorpora son las siguientes:

- La pulsación de la tecla ↑ muestra en la línea de mandatos el último mandato que se ha ejecutado. Si se pulsa sucesivamente esta tecla se va recorriendo el histórico de mandatos hacia atrás y pulsando ↓ se avanza en sentido contrario.
- La pulsación de las teclas ← y → permiten moverse a lo largo de la línea de mandato para modificarlo.
- La pulsación de la tecla ^D permite eliminar el carácter situado bajo el cursor.
- La pulsación de la tecla ^E avanza el cursor hasta el final de la línea.
- La pulsación de la tecla ^A sitúa el cursor al principio de la línea.
- La introducción del mandato vacío (“return”) provocará que se vuelva a ejecutar el último mandato.

5. TUTORÍAS DE LA PRÁCTICA

Las posibles preguntas relacionadas con la práctica se atenderán por correo electrónico, dirigido a pr_mcach@dati.fi.upm.es, o personalmente. en el horario de tutorías de las profesoras encargadas de la práctica.

Se recomienda a los alumnos que, **especialmente fuera del periodo lectivo** (meses de julio y septiembre, navidad y semana santa) establezcan una cita con el profesor al que desean consultar mediante correo electrónico.

6. NORMAS DE PRESENTACIÓN

La práctica se realizará **individualmente** en cualesquiera de las **dos convocatorias disponibles: febrero y septiembre** (para todos los alumnos matriculados); y **junio** (*sólo para alumnos repetidores matriculados*).

Se recomienda a los alumnos que consulten periódicamente la sección de noticias del entregador de la práctica y, también, su hoja web. En ambos lugares se publicarán novedades y avisos relacionados con la práctica.

CONVOCATORIA DE FEBRERO

Desde el **12** hasta el **21** de diciembre de 2011, y desde el **9** de enero hasta el **24** de enero de 2012, ambos inclusive, se realizarán correcciones de la práctica todos los **días hábiles** a las 21:00. De estas, el alumno podrá disponer de **tres** correcciones.

El día **26 de enero de 2012** se realizará una última corrección extraordinaria a la que podrán presentarse todos los alumnos, por lo que disponen de (3 + 1) correcciones en total.

Para solicitar una corrección es suficiente con realizar correctamente la entrega de los ficheros de la práctica (apartado **6.1. Entregas de la práctica**).

El cuestionario de la práctica, se entregará una vez concluido el período de entrega de los ficheros de la práctica y en cualquier caso antes del día **30 de enero de 2012 a las 21:00**, tal como se indica en el apartado **6.1.2. Entrega del cuestionario**.

El examen de esta práctica se realizará el día 30 de Enero, tras el examen final de la asignatura, en las aulas que se anunciarán en su momento.

CONVOCATORIA DE JUNIO

A esta convocatoria sólo podrán presentarse aquellos alumnos que estén repitiendo matrícula.

Desde el **14 de mayo** hasta el **8 de junio de 2012**, ambos inclusive, se realizarán correcciones de la práctica todos los días hábiles a las 21:00. De éstas, el alumno podrá disponer de **tres** correcciones. El día **11 de junio de 2012** se realizará una última corrección extraordinaria, a la que podrán presentarse todos los alumnos, por lo que disponen de (3 + 1) correcciones en total.

Para solicitar una corrección es suficiente con realizar correctamente la entrega de los ficheros de la práctica (apartado **6.1. Entregas de la práctica**).

El cuestionario de la práctica, se deberá entregar antes del día **12 de junio de 2012 a las 21:00**.

El examen de esta práctica se realizará el 12 de junio de 2012 tras el examen final de la asignatura, en las aulas que se especificarán en su momento.

CONVOCATORIA DE SEPTIEMBRE

Desde el **4 de julio** hasta el **26 de julio** y el **3 de septiembre** se realizarán correcciones de prácticas todos los días hábiles a las 21:00. De éstas, el alumno podrá disponer de **tres** correcciones. El día **4 de septiembre de 2012** se realizará una última corrección extraordinaria a la que podrán presentarse todos los alumnos, por lo que disponen de (3 + 1) correcciones en total.

Para solicitar una corrección basta con realizar correctamente la entrega de los ficheros de la práctica (apartado **6.1. Entregas de la práctica**).

El cuestionario de la práctica, se deberá entregar antes del día **5 de septiembre de 2012 a las 21:00**.

El examen de esta práctica se realizará el 5 de septiembre, tras el examen final de la asignatura, en las aulas que se especificarán en su momento.

Durante el mes de Agosto no habrá tutorías ni atención directa a los alumnos, por lo que si surgen problemas no podrán resolverse hasta el mes de septiembre.

6.1. ENTREGAS DE LA PRÁCTICA

6.1.1. Entregas de los ficheros

En cualquier entrega de la práctica se deben suministrar **obligatoriamente** los siguientes ficheros:

- **autores:** Es un fichero ASCII que deberá contener los apellidos, nombre, número de matrícula y DNI de los autores de la práctica. La práctica se realizará individualmente o en grupos de **dos alumnos** (véase el apartado anterior). 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; 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):

- **cuestionario.txt**: Es un fichero que contendrá una copia ASCII del cuestionario. Éste se puede escribir con cualquier editor de textos y se salvará en ASCII para su entrega. En caso de no tener cumplimentado el cuestionario en el momento de hacer una entrega, se permite que este fichero se entregue vacío. Sin embargo, se aconseja realizar la entrega del fichero cumplimentado tan pronto como sea posible.
- Ficheros de código:
 1. **ej1_inter.ens**
 2. **ej1_unroll.ens**
 3. **ej2_fus.ens**
 4. **ej2_rees.ens**

6.1.2. Entrega del cuestionario

La última entrega de la práctica se refiere únicamente al cuestionario y se compone de:

1. El fichero **cuestionario.txt** en ASCII, debidamente completado.
2. Una copia impresa del **cuestionario**, debidamente cumplimentado, en formato DIN-A4, en cuya primera página deberán figurar claramente el nombre y apellidos de los autores de la práctica. Este cuestionario se deberá depositar en **el buzón del departamento** (bloque 4, planta 1).

FORMA DE ENTREGA DE LOS FICHEROS

Alumnos que hayan realizado la práctica en *batman*.

Se utilizará un programa de entrega denominado **ent_cach**. Para ejecutar este programa se deberá teclear, desde el intérprete de comandos de **batman**, la palabra **ent_cach**. Dicho programa, permite entregar los ficheros indicados anteriormente, así como consultar los resultados de la ejecución de un conjunto de tests de pruebas utilizados 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):

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

Introduzca su identificador (Num. matricula): A990999

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 la operación se ha completado 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:
990999 123433342 PEREZ PEREZ JESUS**

Se ha asignado password al usuario A990999.

- **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

Si dicho alumno no aparece en las listas en poder del 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 (los nombres de los ficheros son genéricos, no necesariamente corresponden a los que debe entregar en esta convocatoria):

MANDANDO EL FICHERO reesc.ens ...OK.

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

MANDANDO EL FICHERO fusion.ens ...
No se puede abrir el fichero fusion.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 cuando se realice la corrección. Si el ensamblado no ha finalizado con éxito se mostrará un mensaje:

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

En este caso el alumno deberá comprobar que se puede ensamblar correctamente el fichero y comprobar 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 la ejecución del programa 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 grupos cuyas prácticas están 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 de fichero en el que se copiarán los resultados de la ejecución del conjunto de *tests* de pruebas que componen el corrector. El programa pedirá un nombre de fichero donde escribir los datos generados. 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 teclaea 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. Esta es la razón por la que los casos de prueba utilizados para la corrección de la práctica no se ponen a disposición del alumnado.

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 tecldea 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 SERÁ POSIBLE CORREGIR NINGUNA PRÁCTICA QUE NO SE ATENGA A ESTAS NORMAS Y SE CONSIDERARÁ POR LO TANTO COMO NO PRESENTADA.

Alumnos que no hayan realizado la práctica en *batman*.

Aquellos alumnos que realicen la práctica sin utilizar los medios del Centro de Cálculo (utilización de PC's con Linux) pueden realizar la entrega de la práctica desde la red de PC's de la Facultad. Para ello se deben situar en la unidad de trabajo donde tengan los ficheros de la práctica y teclear:

```
G:\>a:
A:\>cd practica
A:\PRACTICA>G:\DATSI\CACHES\ENT_CACH
```

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 práctica. ¡Se recomienda trabajar siempre sobre disquete y nunca sobre disco duro!

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

PROCEDIMIENTO DE ENTREGA VÍA WEB

Se ha desarrollado una aplicación Web que permite realizar las mismas operaciones que la aplicación descrita en el apartado anterior. La URL en la que se encuentra es:

`http://www.datsi.fi.upm.es/Practicas`