ESTRUCTURA DE COMPUTADORES

Departamento de Arquitectura de Computadores

Relación de Problemas del Tema 3: Jerarquía de Memoria.

1. Una computadora tiene un bus de direcciones de 32 bits y un direccionamiento a nivel de byte. La caché tiene una capacidad de 256 Kbytes, con palabras de 1 byte y bloques de 32 bits organizados con la poliítica de organización directa, reemplazo LRU implementado con 5 bits, y post-escritura usando un bit de `dirty` para los bloques modificados pendientes de ser escritos en memoria principal.

En base a estas premisas, responder a las siguientes cuestiones:

a) ¿Cuántos bits tiene el campo etiqueta (TAG) de cada entrada del directorio caché?

b) ¿Qué valores tienen los parámetros `m` y `c`?

c) ¿Qué tamaño en bits ocupa el directorio caché al completo? Considerar los datos, las etiquetas y los bits de reemplazo y escritura para cada una de sus entradas.

2. Una memoria caché asociativa por conjuntos de 2 bloques con direcciones de 32 bits y palabras de 1 byte tiene 4 bytes en cada uno de sus bloques y una capacidad total de 128 Kbytes. El direccionamiento es a nivel de byte.

Se pide:

a) El tamaño de una dirección de memoria y su descomposición en los campos número de línea (N), etiqueta (N-c), conjunto (c) y número de palabra dentro del bloque (m).

b) Indicar el valor de cada campo para las siguientes direcciones de memoria principal: `0284A482h, 01148C89h, 0038CF00h, 0038CF01h`.

c) ¿Pueden todos los bloques que incluyen las referencias anteriores estar en caché al mismo tiempo?

3. Dado un sistema de memoria principal de 1 Mbyte con palabras de 1 byte y un programa que referencia a la siguiente secuencia de direcciones (en hexadecimal):

`ABC80h, ABC81h, ABC88h, BCD90h, BCD9Dh, BCDA0h, CDE00h, CDE18h, CDE20h`

Considerar las dos estrategias que se detallan a continuación para acelerar el acceso a memoria principal:

I) Entrelazar la memoria principal siguiendo un esquema de orden inferior con 32 módulos.

II) Incorporar una memoria caché de 64 Kb. organizada asociativamente en 4 conjuntos, 8 palabras por bloque y algoritmo de reemplazo FIFO.

Se pide:

a) Analizar y comparar el tiempo total de respuesta (en ciclos) de esos dos sistemas de memoria para la secuencia de direcciones solicitada por el programa. Considerar una latencia de 10 ciclos para el acceso a un banco DRAM de la memoria principal y que la transferencia de datos por el bus de memoria desde los módulos DRAM a la caché tarda 1 ciclo. Considerar así mismo que se tarda 1 ciclo en consultar el directorio caché y suministrar el dato si hay acierto, mientras que en caso de fallo habrá que considerar el tiempo de transporte del bloque desde la memoria a la cache.

   Suponer la caché inicialmente vacía e ilustrar claramente el contenido del directorio cache tras cada acceso a memoria principal.

b) Discutir posibles mejoras en ambos sistemas y su repercusión en el coste: En el entrelazado, respecto al número de módulos; en la caché, respecto a su organización y reemplazo de líneas.

4. Disponemos de una memoria caché de 4 Kb. con tamaño de bloque de 256 bytes, organización asociativa de 8 conjuntos y algoritmo de reemplazo LRU.

Conectamos la caché entre un procesador de 8 bits y una memoria principal de 1 Mbyte, siendo el byte la unidad direccionable. Se pide:

a) Suponiendo la caché inicialmente vacía, describir la evolución del directorio caché para la siguiente secuencia de petición de direcciones a memoria principal:

   `319F0h, 31AF0h, 7013Ch, 77777h, 44037h, 778DEh, A5021h`

b) Comparar el sistema anterior con respecto a criterios de rendimiento y coste frente a una caché organizada de forma directa y a otra totalmente asociativa.

5. Un adicto a los PCs pretende evaluar el comportamiento de la caché L1 de datos de un PC para el famoso juego Tomb Raider III. Dispone para ello de una arquitectura Pentium con 16 Mbytes de memoria principal. Recordaremos que el Pentium direcciona la memoria a nivel de byte, y que dispone de una caché L1 interna de 8 Kbytes para datos. En cuanto al Tomb Raider, comienza con un bucle de 100 iteraciones, referenciando en cada una de ellas a una secuencia de 4 datos ubicados en las siguientes direcciones (bytes) de memoria principal (en hexadecimal): `000A40h, 004A40h, 008A40h, 00BA40h`.

Nuestro amigo observa que la caché de datos evoluciona de la siguiente manera para la primera iteración del juego (en las 99 restantes, el comportamiento es muy similar):

Al final de la primera referencia:

C	L	Etiq
0	0	-
	1	-
1	0	-
	1	-
2	0	-
	1	-
3	0	-
...	...	-
81	1	-
82	0	000h
	1	-
83	0	-
	1	-
84	0	-
...	...	-
126	1	-
127	0	-
	1	-

Al final de la segunda referencia:

C	L	Etiq
0	0	-
	1	-
1	0	-
	1	-
2	0	-
	1	-
3	0	-
...	...	-
81	1	-
82	0	000h
	1	004h
83	0	-
	1	-
84	0	-
...	...	-
126	1	-
127	0	-
	1	-

Al final de la tercera referencia:

C	L	Etiq
0	0	-
	1	-
1	0	-
	1	-
2	0	-
	1	-
3	0	-
...	...	-
81	1	-
82	0	008h
	1	004h
83	0	-
	1	-
84	0	-
...	...	-
126	1	-
127	0	-
	1	-

Al final de la cuarta referencia:

C	L	Etiq
0	0	-
	1	-
1	0	-
	1	-
2	0	-
	1	-
3	0	-
...	...	-
81	1	-
82	0	008h
	1	00Bh
83	0	-
	1	-
84	0	-
...	...	-
126	1	-
127	0	-
	1	-

El significado de cada campo es el siguiente: C = Número de conjunto; L = Número de línea; Etiq = Campo etiqueta del directorio caché; "-" = Línea vacía).

a) En función del comportamiento mostrado en los diagramas, se pide obtener el número de conjuntos de su caché de datos, el número de bits utilizado para las etiquetas del directorio caché y el tamaño de la línea de caché del Pentium (en bytes), así como todas las estrategias de reemplazo de líneas que pueden dar lugar a la planeación anterior (excluyendo RANDOM). Nota: Todos los parámetros utilizados en el problema (`w, c, n` y `N`) coinciden fielmente con la implementación comercial.

N= [ ] c= [ ] n= [ ] w= [ ]

Reemplazo: [ ]

b) Nuestro amigo quiere ahora cambiar el microprocesador de su equipo, dudando entre adquirir un Pentium MMX o un K6-2.

El MMX dispone de 16 Kbytes para la caché de datos, donde se respetan las mismas características que en el Pentium original, aumentando únicamente el nivel de asociatividad (esto es, se tiene el mismo número de conjuntos, pero se dobla el número de líneas por conjunto).

El K6-2, por el contrario, tiene una caché de datos de 32 Kbytes, pero mantiene el nivel de asociatividad y el resto de parámetros del Pentium original, con la salvedad del número de conjuntos, que es ahora cuatro veces superior.

¿Qué microprocesador recomendarías a nuestro amigo atendiendo exclusivamente al índice de aciertos a caché de datos producido por las referencias a memoria efectuadas desde el mencionado bucle del Tomb Raider III? (Ayúdate de una traza del directorio caché para cada caso con el mismo formato que el que hemos utilizado para los diagramas de la caché del Pentium).

6. Considérese un procesador con instrucciones de 32 bits y una caché de instrucciones de 32 bytes con bloques de 8 bytes. Para los dos fragmentos de código MIPS siguientes, indicar la organización (y sus parámetros) que dá lugar a una ejecución con el mínimo número de fallos (si hay varias que produzcan el mismo rendimiento, deberá escogerse la de menor coste hardware).

```
0  lw  $1, 100($2)
1  add $1, $1, $3
2  sub $4, $5, $1
3  j   8
...
6  mul $2, $2, $8
7  sub $2, $2, $9
8  div $8, $1, $4
9  j   0
```

```
0  lw $1, 100($2)
1  add $1, $1, $3
2  sub $4, $5, $1
3  j   8
...
6  mul $2, $2, $8
7  sub $2, $2, $3
8  add $8, $1, $0
9  j   6
```

7. Se quiere ejecutar el siguiente programa en un procesador MIPS:

```
        #0  lw  $1, %2
suerte: #1  sw  $1, %10
        #2  j   animo
        ...
animo:  #35 sw  $0, %2
        #36 sw  $2, %26
        #37 lw  $3, %42
        #38 lw  $1, %50
        #39 bne $1, $0, suerte
```

donde se supone que la instrucción `bne` salta la primera vez que se ejecuta y no lo hace la segunda, momento en que termina el programa.

Las instrucciones se han numerado precedidas por "#" según su orden de ubicación en memoria, comenzando en la dirección 0 (por ejemplo, #35 significa "instrucción número 35", nunca "dirección 35 de memoria"). Los operandos precedidos por "%" simbolizan una petición de datos, que para simplificar la resolución del problema indican ya directamente el número absoluto de línea de memoria principal en la que se encuentra ese dato. Los números precedidos por "$" corresponden a registros del banco como ya es costumbre. Independientemente de su prefijo, ("#", "%" o "$"), todos los números están representados en formato decimal.

Implementamos el primer nivel de memoria del MIPS mediante dos cachés internas (L1), una para instrucciones con organización directa, y otra para datos asociativa de 4 conjuntos con reemplazo LRU. Ambas cachés son de 128 bytes, con un tamaño de bloque de 8 bytes.

Recordando que el MIPS es un procesador de 32 bits y que su bus de direcciones es también de 32 bits, queda claro que en cada bloque de caché caben dos instrucciones (o dos datos).

Para cada caché por separado, se pide:

a) Indicar la secuencia de bloques solicitados por parte del programa MIPS.

b) Describir la composición del directorio caché. Mostrar su evolución para la secuencia de peticiones anterior, indicando claramente fallos y reemplazos donde corresponda.

c) Calcular el índice de fallos.

8. Sea el siguiente código MIPS:

```
      ori  $2, $0, 1000h
loop: lw   $1, 2800h($2)
      sub  $4, $1, $0          rotar: add  $10, $4, $4
      jal  rotar                      muli $7, $10, 2
      sw   $7, 7800h($2)              jr $31
      sw   $1, C800h($2)
      subi $2, $2, 4
      bne  $2, $0, loop
```

Se pide:

a) Partiendo de que el procesador tiene 32 bits de tamaño de palabra, responder a las siguientes cuestiones (recordar además que `1000h` es 4K (ó 4*1024), `2800h` es 10K, `7800h` es 30K y `C800h` es 50K):

1) ¿Qué tamaño de bloque recomendarías para servir las peticiones a memoria de datos cursadas por el programa sobre una caché de 4 Kbytes organizada de forma directa?

2) Suponiendo que nos piden diseñar el sistema con tamaño de bloque de 256 bytes, ¿se podría evitar algún fallo reduciendo el número de conjuntos?

3) ¿A partir de cuántos bloques por conjunto no tiene sentido seguir aumentando el nivel de asociatividad de la caché anterior?

4) Para la configuración que has elegido en el apartado anterior, ¿Varían algo los resultados si cambiamos de reemplazo FIFO a LRU o viceversa?

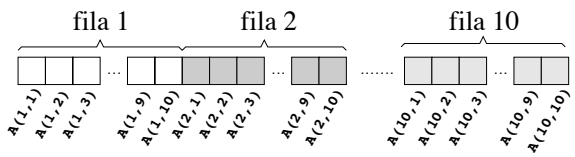
9. En la figura se proponen dos códigos en alto nivel que inicializan ambos una matriz de 10×10 números reales a cero. El código generado por el compilador almacena la matriz en la memoria en posiciones consecutivas por filas, tal como se muestra en la figura adjunta. Las variables i, j son ubicadas en dos registros diferentes del procesador.

```
i=1
While (i<=10) Do
  j=1
  While (j<=10) Do
     A(i+1,j)=0.0
     A(i,  j)=0.0
     j=j+1
  End While
  i=i+2
End While
```

Código (A)

```
i=1
While (i<=10) Do
  j=1
  While (j<=10) Do
     A(j,i)=0.0
     j=j+1
  End While
  i=i+1
End While
```

Código(B)



Se dispone de un sistema de memoria cuya caché es completamente asociativa, con reemplazo FIFO. El tamaño de la caché es equivalente a cinco números reales en la representación usada por el procesador, y permite almacenar un número por bloque.

a) Ambos códigos realizan la misma función, pero desde el punto de vista del sistema de memoria el rendimiento es diferente. Analiza cuál de ellos daría más fallos de caché atendiendo a sus localidades espaciales.

10. Queremos simular un simple bucle como éste:

```
DO i=0,9
  A(i)=0
ENDDO
```

en una máquina con una memoria cache de 8 bytes, con tamaño de bloque de 2 bytes. Sabiendo que `A` es un array de 30 números almacenados en memoria a partir de la posición 0, y que cada elemento de ese array ocupa 1 byte:

a) ¿Qué tipo de organización de memoria cache (directa, por conjuntos o totalmente asociativa) elegirías en función del porcentaje de fallos? Nota: a igualdad de rendimiento se debe elegir la menos costosa desde el punto de vista HW). Justifica la respuesta.

b) ¿Se podría decir lo mismo independientemente de la posición de comienzo del array `A` en memoria?. Razonar.

Imaginemos ahora que el bucle que estamos procesando es este otro (se supone que en la sentencia `A(i)=B(9-i)` el acceso de lectura ocurre antes que el de escritura):

```
DO i=0,9
  A(i)=B(9-i)
ENDDO
```

donde `A` es el mismo array anterior, y `B` es otro array de 45 números de 1 byte, almacenado a partir de la posición 60 de memoria.

a) ¿Qué tipo de organización de memoria cache elegirías ahora?. Justifica la respuesta calculando el porcentaje de fallos para cada caso.

b) ¿Es independiente la respuesta anterior de la posición de comienzo del array B en memoria?. Pon ejemplos explicativos.