Pontificia Universidad Católica de Chile
Escuela de Ingeniería
Departamento de Ciencia de la Computación

IIC2343 – Arquitectura de Computadores (II/2012)

Solución Interrogación 3

Pregunta 1

a) Explique y describa el concepto de jerarquía de memoria. Describa un esquema en que se use una jerarquía de tres niveles y detalle cada uno de estos. **(1 pto.)**

Solución: La jerarquía de memoria de un computador se basa en el uso de diversos niveles de memoria, cada uno de los cuales ocupa una tecnología distinta, aprovechando las ventajas de cada tecnología: para almacenar mucha información se utilizan tecnologías que tienen un costo bajo por Byte, como los discos magnéticos; para acceder rápidamente a información se utilizan tecnologías electrónicas de rápido acceso, como SRAM (static RAM); para niveles intermedios se utilizan tecnologías con velocidades de acceso y costos intermedios, como DRAM (dynamic RAM).

b) En un computador con arquitectura x86, un dispositivo defectuoso genera interrupciones aleatorias que degradan fuertemente el rendimiento del sistema. ¿Es posible eliminar este problema sin desconectar el dispositivo? **(1 pto.)**

Solución: Si, es posible hacerlo mediante el enmascaramiento de interrupciones del PIC.

c) ¿Qué ventaja tiene una caché del tipo **split** sobre una **unified**? **(1 pto.)**

Solución: La caché split tiene zonas independientes para datos e instrucciones, por lo que puede aprovechar de mejor manera los patrones de localidad espacial y temporal de datos e instrucciones, que la gran mayoría de las veces son distintos.

d) Un fabricante de hardware sacará prontamente al mercado una tarjeta de video (GPU) con 4 GB de RAM. ¿Qué mapeo de memoria le recomendaría al fabricante para poder interactuar con este dispositivo? **(1 pto.)**

Solución: Se le recomienda utilizar direcciones de memoria independientes para sus registros de comandos y estado, y además mapear sólo un parte de la memoria de video a la memoria principal (para no saturar esta) y dejar que la tarjeta de video se encargue y DMA se encarguen de completar las transferencias.

e) Dada un memoria caché de 1 KB con bloques de 16 palabras de 1 byte y función de correspondencia **fully associative**, que se encuentra llena, ¿cuantas comparaciones secuenciales se deben realizar para decidir que bloque sustituir? Describa además un esquema que permita minimizar el tiempo requerido para este proceso, usando la misma cantidad de comparaciones. **(1 pto.)**

Solución: La caché tiene 64 bloques, por lo que serán necesarias 63 comparaciones secuenciales. Si estas se hacen se manera paralela, de 2 en 2, se realizan las mismas 63, pero demorarían aproximadamente $frac110$ del tiempo.

f) Describa una posible solución al problema de consistencia entre caché y RAM del esquema **write-back**. **(1 pto.)**

Solución: Es posible solucionar esto agregando un bit de consistencia a cada bloque de la memoria RAM. Si el bloque ha sido modificado en la caché y no en la RAM, se setea este bit en 0. Luego, en caso que se requiera utilizar ese bloque por algún elemento del computador (ej. controlador DMA), si el bit de consistencia está en 0, deberá actualizarse el bloque en la memoria RAM.

Pregunta 2

a) Considere una jerarquía de memoria dada por memoria RAM y caché, donde ambas utilizan palabras de 16 bits (16 bits = 2B = W). La memoria RAM tiene 1 GW (G = 1 Giga = 1024*1024*1024) de capacidad, mientras que la caché una capacidad 1 MW (M = 1 Mega = 1024*1024) y bloques de 256W. Responda las siguientes preguntas, asumiendo que la caché utiliza una función de correspondencia **4-way associative**:

- Calcule el número de bloques y conjuntos de la memoria caché. **(0.5 ptos.)**

  Solución: La cachés de 1 MW y tiene bloques de 256W, por lo tanto tiene $\frac{1024 \times 1024}{256} = 4096$ bloques. Luego, si la caché es **4-way associative**, tiene $\frac{4096}{4} = 1024$ conjuntos.

- Describa la división de la direcciones de memoria, incluyendo offset dentro del bloque, conjunto y tag. **(0.5 ptos.)**

  Solución: La memoria principal es de 1 GW, por lo tanto tiene 2^{30} palabras de 16 bits, luego, las direcciones de memoria tienen 30 bits. Como los bloques son de 256W y tenemos además 1024 conjuntos, necesitamos los 8 bits menos significativos de la dirección para el offset y los 10 bits siguientes para el conjunto. Por lo tanto, los 12 bits más significativos se utilizan para el tag.

b) Considere un computador con microarquitectura Von Neumann, donde la tasa de ciclos de clock por instrucción es igual a **1.0** cuando todos los accesos a memoria producen hits en la caché. Además, esta memoria caché tiene un **miss rate** de 2 % y **miss penalty** de 25 ciclos de clock. Si en un programa el 50 % de las instrucciones corresponde a lectura/escritura de memoria, ¿cuánto más rápida sería la ejecución del programa si todas las instrucciones del programa produjeran hits en la caché? **(2 ptos.)**

Solución: Si se asume un programa de 100 instrucciones, sabemos que cuando el hit-rate es 100 %, el programa demorará 100 ciclos. Si tenemos ahora una situación donde el hit rate es 98 %, como en la arquitectura Von Neumann cualquier instrucción requiere al menos un acceso a memoria (fetch), se tiene que 2 instrucciones, de un total de 100, tendrán un caché miss. Además de esto, se sabe que el 50 % de las instrucciones son de escritura/lectura, por lo tanto, como el 98 % de 50 es 1, tendremos 1 instrucción que generará un miss cuando quiera acceder los datos. Sumando, tenemos 97 instrucciones con hit y 3 con miss, por lo que el tiempo total será de 175 ciclos, por lo que la ejecución cuando el hit-rate es 100 % será 1.75 veces más rápida.

c) Un computador de 32 bits tiene una caché, sólo para datos, de 32 KB con bloques de 64 palabras de 1 byte y función de correspondencia **2-way associative**, además de política de sustitución **LFU**. Considere el siguiente programa:

```
int m[512][512];
int sum = 0;
for (int i = 0; i < 512; i ++)
{
    for (int j = 0; j < 512; j++)
    {
        sum += m[i][j];
    }
}
for (int i = 0; i < 512; i ++)
{
    for (int j = 0; j < 512; j++)
    {
        sum += m[j][i];
    }
}
```

Asumiendo que la variable *sum* y los contadores i y j se almacenan en registros y que la matriz m se almacena en memoria en orden de filas, calcule el hit rate que produce el programa anterior en la caché. **(3 ptos.)**

Solución: De los datos del enunciado, se puede extraer que la caché tiene 512 bloques y 256 conjunto. Además, como un dato de tipo *int* utiliza 4 bytes (32 bits), cada bloque podrá contener 16 ints. De la misma manera, para almacenar una fila de la matriz se necesitarán 32 bloques de la caché, mientras que para llenar la caché se necesitan sólo 16 filas de la matriz. Teniendo esto en consideración y sabiendo que la matriz es almacenada por filas, se puede notar que 1 de cada 16 lecturas en el primer **doble-for** producirá un miss, mientras que las otras 15 serán hits, ya que el bloque (16 ints) estará almacenado en la caché debido al miss previo. Por lo tanto, el hit-rate luego del primer **doble-for** es de $\frac{15}{16}$. Para el segundo **doble-for** la situación es distinta, pero más simple. Dado el tamaño de la matriz y la distribución de conjuntos y bloques en esta, el recorrido en orden de columnas no generará nunca un hit en la caché, ya que los bloques escritos en esta serán sustituidos antes de poder ser accedidos nuevamente. Por lo tanto, en este caso el hit-rate es 0. Luego, si sumamos los dos casos, tenemos que el hit-rate total es $\frac{15}{16} + \frac{0}{16} = \frac{15}{32}$.

Pregunta 3

Suponga que se tiene un dispositivo de adquisición de imágenes térmicas conectado a un computador que tiene una microarquitectura especializada para a la adquisición de imágenes, pero con ISA compatible con x86 de 16 bits. El computador tiene una memoria principal de 64 kilobytes, con el siguiente mapa de memoria para los primeros 4096 bytes:

Dirección	Función asociada
0-5	Exception handlers
6	Registro de comandos de la cámara
7	Registro de estado de la cámara
8-14	Vectores de interrupciones de hardware
15	Vector de interrupción de escritura en disco
16	Vector de interrupción de adquisición de imagen
17-31	Vectores de interrupciones de software de uso libre
32-123	Memoria de uso libre
124-1023	Buffer de adquisición de la cámara.
1024-4096	Espacio de memoria del disco.

Se desea escribir un programa que permita adquirir imágenes mediante la cámara y luego almacenarlas en disco. Las imágenes generadas por la cámara se encuentran en escala de grises de 8 bits, ordenadas por filas en una matriz cuadrada de 30x30.

a) Escriba una ISR para alguna interrupción de software disponible, que permita adquirir una imagen y luego escribirla en disco.
La ISR de la cámara no recibe parámetros y retorna en su registro de estado información sobre la adquisición. Si la adquisición fue exitosa, el registro contendrá 0xFF y la imagen se encontrará en el buffer de la cámara. En caso contrario, si la adquisición falló, el registro contendrá 0x00. Durante la adquisición, el registro contendrá el valor 0xF0.
La ISR del disco utiliza internamente el controlador de DMA, por lo que necesita los siguientes parámetros en los siguientes registros:

- La dirección de inicio del origen en el registro AX.
- La dirección de inicio del destino en disco en el registro BX.
- La cantidad de palabras a copiar en el registro CX.

Puede utilizar la cantidad de parámetros que estime conveniente para su ISR, pero debe dejar explícitamente escrito qué significan y donde se almacenan. **(4 ptos.)**

Solución: La solución utiliza la IRQ 17 y asume que recibe como parámetro la dirección de inicio de escritura en disco en el registro BX.

```
ISR17:
    INT 16
while:
    MOV AX, [7]
    CMP AX, F0h
    JE while

    MOV AX, 124
    MOV CX, 900
    INT 15
```

b) Escriba un programa que llame a la subrutina del item anterior para adquirir tres imágenes y almacenarlas de manera consecutiva en disco. Considere que la adquisición puede fallar y que se intentará esta un máximo de tres veces por imagen. **(2 pto.)**

Solución: La solución utiliza la IRQ 17 y asume que recibe como parámetro la dirección de inicio de escritura en disco en el registro BX.

```
        MOV DX, 0
        MOV BX, 1024
        MOV SI, 0
    while:
        CMP DX, 3
        JE end
        ADD DX, 1
        INT 17
        MOV AX, [7]
        CMP AX, 00h
        JE while
    end:
        MOV DX, 0
        ADD BX, 900
        ADD SI, 1
        CMP SI, 3
        JLT while
```