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

IIC2343 – Arquitectura de Computadores (II/2015)

Interrogación 3

Pregunta 1

a) ¿Que mecanismo de hardware se utiliza para informar a un proceso que la copia de memoria que solicitó ya fue terminada? **(1 pto.)**
Solución: Se utiliza una solicitud de interrupción generada por el controlador de DMA.

b) Cuando se atiende una solicitud de interrupción (IRQ), ¿en que momento se deben respaldar los flags de la CPU y por qué? **(1 pto.)**
Solución: Se deben respaldar inmediatamente después de terminar de ejecutar la instrucción actual. Si se hace después, se perderán los flags debido a la ejecución de otras instrucciones.

c) Un computador compatible con el assembly x86 de 16 bits, tiene conectado un controlador de DMA mediante los siguientes puertos:

Puerto	Función
33	Comandos
34-35	Origen
36-37	Destino
38-39	Cont. palabras

Este controlador inicia la copia cuando recibe su registro de comandos la palabra 0x01 y puede utilizarse sólo mediante la interrupción de software 22. Además, este computador cuenta con un timer programable, que es capaz de informar a la CPU cuando hayan transcurrido X ms, mediante una IRQ. El timer se activa enviado la palabra 0x01 al puerto 53, mientras que el tiempo para la interrupción se programa enviando un número natural de 8 bits al puerto 54. Finalmente, ISR_TIMER es la subrutina que maneja la interrupción.

i. Escriba el código de ISR_DMA, que corresponde a la subrutina que se ejecuta cuando se llama a la interrupción de software 22. Defina claramente la convención de llamada que usará la ISR. **(1 pto.)**
Solución: Asumimos que la convención de llamada para esta interrupción de software, consiste en almacenar en AX la dirección de origen, en BX la dirección de destino y en CX la cantidad de palabras.

```
OUT 34, AL
OUT 35, AH
OUT 36, BL
OUT 37, BH
OUT 38, CL
OUT 39, CH
OUT 33, 0x01
```

II. Asuma que el computador ejecuta un sistema operativo con soporte para multiprogramación que implementa *preemptive scheduling*. Teniendo esto en cuenta, escriba el código de ISR_TIMER, que se encargará de respaldar el PCB del proceso saliente, cargar el del proceso entrante y de programar el timer para que genere una nueva interrupción, donde cada PCB tiene un tamaño de 1KB. El PCB del proceso en ejecución se almacena a partir de la posición de memoria 0x0100. Asuma que los parámetros para realizar el cambio de contexto (dir. origen PCB entrante, dir. destino PCB saliente, tiempo para siguiente interrupción), se encuentran disponibles, en el mismo orden, a partir de la dirección de memoria 0x0000. **(3 ptos.)**

Solución:

```
ISR_TIMER:
  MOV AX, 0x0100
  MOV BX, 0x0002
  MOV CX, 0x0400 ; 0x0400 = 1024
  INT 22

  MOV AX, 0x0000
  MOV BX, 0x0100
  INT 22

  MOV AL, [0x0004]
  OUT 54, AL
  OUT 53, 0x01
  IRET
```

Asignación de puntaje: 1 pto. por cada copia de PCB, 0.5 ptos. por programar el timer, 0.5 ptos por terminar correctamente la ISR (IRET).

Pregunta 2

a) Un ISP (Internet Service Provider) tiene un ping de 7 ms y respalda diariamente el 30 % de los recursos requeridos por los usuarios, para acelerar el acceso. En caso de no tener respaldado un recurso, toma X segundos en obtenerlo. ¿Cuál es el tiempo de acceso promedio a un recurso cualquiera en la red, asumiendo que ese recurso también fue requerido el día anterior? **(1 pto.)**
Solución: $TAP = 7 + 0{,}7 \times 1000X = 707\ ms$.

b) ¿Por qué es importante que algunos niveles de memoria caché sean exclusivos para cada núcleo y otros compartidos? **(1 pto.)**
Solución: Es muy posible que cada núcleo ejecute programas distintos, por lo tanto es necesario que los datos e instrucciones que usan se mantengan almacenadas de manera independiente a las de otro núcleo. Así, estas no tienen conflicto por el espacio en la memoria caché. Por otro lado, los programas necesitan generalmente compartir datos, como por ejemplo los parámetros para llamadas a subrutinas. Así, para evitar usar la memoria principal (más lenta), para compartir estos datos, se utiliza una memoria caché ubicada en la parte más bajo de la jerarquía.

c) Las aplicaciones que reproducen audio o video son parte de un tipo de aplicaciones que generan un trabajo en la memoria del tipo *streaming*, que consiste en la lectura de grandes cantidades de datos, pero una muy baja reutilización de estos. Para este ejercicio considere un computador que corre una aplicación que realiza *streaming* para acceder a 512KB de datos. El patrón de acceso a memoria es el siguiente: 0, 4, 8, 12, 16, ...

I. Asuma que el computador tiene una memoria caché de 64 KB con mapeo directo y líneas de 32 bytes. ¿Cuál es el *miss-rate* para el patrón de accesos a memoria descrito anteriormente? ¿Cuán sensible es este *miss-rate* con respecto al tamaño de la memoria caché? ¿Mejora el *hit-rate* si se utiliza una caché 4-way associative? **(1 pto.)**
Solución: Dado que 32 es un múltiplo de 4, se realizan 8 lecturas por bloque/línea. Dado que el patrón de accesos es estrictamente ascendente, el miss rate en un bloque es igual al de toda la caché, sin importar el tamaño de esta. Por lo tanto, el miss-rate es $\frac{1}{\frac{32}{4}} = 12{,}5\,\%$, independiente si la caché usa mapeo directo o 4-way associative.

II. Recalcule el *miss-rate* tomando los siguientes tamaños de línea en una caché con mapeo directo: 16 bytes, 64 bytes, 128 bytes. ¿Qué tipo de localidad aprovecha este tipo de aplicaciones? **(1 pto.)**
Solución: $\frac{1}{\frac{16}{4}} = 25\,\%$, $\frac{1}{\frac{64}{4}} = 6{,}25\,\%$, $\frac{1}{\frac{128}{4}} = 3{,}125\,\%$. Dados estos resultados, queda claro que las aplicaciones de streaming aprovechan la localidad espacial.

III. El *prefetching* es una técnica que saca provecho de patrones de lectura predecibles, para traer desde la memoria de manera especulativa, bloques de memoria adicionales cuando una línea de la caché en particular es accedida. Un ejemplo de esto es el *stream buffering*, que consiste en copiar, en un *buffer* separado, bloques adyacentes a la línea de la caché accedida. Si el dato se encuentra en este *buffer*, se considera como *hit* y es copiado a la caché, mientras que el espacio que este usaba en el *buffer*, es llenado con un nuevo bloque adyacente. Asuma que el computador cuenta con un *stream buffer* de dos entradas y que la latencia de la cache es tal que una línea puede ser cargada antes de que el cálculo realizado sobre la línea anterior ha sido completado. En base a esto, cuál es el *hit-rate* para el patrón de acceso a memoria descrito anteriormente y una caché con líneas de 32 bytes? **(2 ptos.)**
Solución: El hit-rate será casi igual a 1, ya que sólo hay un miss en el primer acceso de memoria. Luego, todas las lecturas encontrarán el dato que necesitan ya sea en la memoria caché o en el stream buffer. Más especificamente, el hit-rate será $\frac{\frac{512\times1024}{4}-1}{\frac{512\times1024}{4}} = 0{,}99999$.

Pregunta 3

a) ¿Cuál es la cantidad mínima de memoria física que debe tener un computador de 32 bits con un esquema de paginación simple como el visto en clases? **(1 pto.)**
Solución: Se necesita como mínimo espacio para una tabla de páginas y para una página.

b) Explique cómo un proceso puede sufrir de inanición si se utiliza el esquema *Fixed priority pre-emptive scheduling*. ¿Es posible que ocurra esto en el esquema *round-robin*? **(1 pto.)**
Solución: Un proceso puede sufrir inanición en el esquema *Fixed priority pre-emptive scheduling* si siempre existen procesos con mayor prioridad. Esto no puede pasar en round-robin, debido a que este esquema despacha los procesos usando FIFO y un tiempo de ejecución fijo.

c) Un computador de 64 bits posee soporte de hardware para paginación, donde cada página es de 4 KB.

I. Describa el proceso que realiza el hardware para obtener el dato buscado, a partir de una dirección virtual. Indique claramente el propósito de cada uno de los bits de la dirección virtual. **(1 pto.)**
Solución: Dado que cada página es de 4KB, se necesitan 12 bits para indicar la palabra dentro de una página. De esta manera, los 52 bits más significativos de la dirección virtual son usados por la MMU para ubicar el marco correspondiente en la tabla de páginas indicada por el registro PRBR. Una vez obtenido el marco, se realiza la petición de memoria usando la dirección fisica, que se construye sustituyendo los 52 bits más significativos de la dirección virtual (número de página), por el número del marco obtenido de la tabla.

II. Si no se ponen restricciones a la cantidad de memoria que puede usar un proceso, ¿cuál es el tamaño de una tabla de páginas (sin contar los bits de validez y disco)? **(1 pto.)**
Solución: Dado que la memoria virtual da acceso a toda la memoria direccionable y que se utiliza un computador de 64 bits, una tabla de páginas tendrá $\frac{2^{64}}{2^{12}} = 2^{52}$ entradas.

III. Proponga un esquema para paginación, que permita que la cantidad de entradas de la estructura que almacena las relaciones marco-página esté acotada por la cantidad de marcos físicos existentes. **(2 ptos.)**
Solución: La solución es utilizar una tabla invertida, que tiene una cantidad de entradas fija e igual a la cantidad de marcos. En cada una de las entradas se indica a que página está asocido el marco. Luego, para obtener un dato desde la memoria, se debe buscar en toda esta tabla la ocurrencia de la página virtual requerida. A pesar de que esto es muy lento, es posible acelerarlo utilizando una tabla de hash.