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

IIC2343 – Arquitectura de Computadores (I/2013)

Solución Interrogación 3

Pregunta 1

a) El principio de localidad espacial explica en parte el buen funcionamiento de la memoria caché. Sin embargo, es posible no cumplir este principio, disminuyendo el rendimiento de la memoria. Describa un ejemplo específico de esto y explique por qué se produce. **(1 pto.)**
Solución: Un ejemplo específico es escribir en memoria una matriz de gran tamaño (que las filas no quepan completamente en la caché) en orden de filas y luego, leer esta matriz en orden de columnas. De esta manera, datos que están espacialmente cercanos en la memoria RAM, no serán leídos en el orden corresondiente.

b) Describa las tres funciones de correspondencia vistas en clases. **(1 pto.)**
Solución:

- Mapeo Directo: Cada bloque de la memoria principal sólo puede almacenarse en un bloque fijo de la memoria caché, definido de acuerdo al tamaño de la caché y el tamaño de los bloques.
- Fully Associative: Cada bloque de la memoria principal puede ubicarse en cualquier bloque de la memoria caché.
- N-way Associative: La memoria caché es dividida en conjuntos de N bloques. Luego, cada bloque de la memoria principal sólo puede ubicarse en un conjunto de la caché, pero en ese conjunto puede ubicarse en cualquier bloque.

c) Describa que elementos de un sistema con soporte para multiprogramación (SO, CPU y Memoria) deben actualizarse al realizar un cambio de contexto. **(1 pto.)**
Solución: Al hacer un cambio de contexto debe actualizarse la TLB, los registros PTBR y de modo de la CPU. En caso de ser necesario, también debe hacerse swap-in de los marcos que hayan sido enviados a disco y deban ser utilizados por el proceso. Finalmente, el PCB del proceso saliente debe actualizarse con los valores de los registros y flags, para poder recuperar su estado en el futuro.

d) ¿Cómo debería modificarse la arquitectura del computador básico para que tenga soporte para multiprogramación? **(1 pto.)**
Solución: Debería agregarse una MMU a la CPU, y los registros PTBR y de modo.

e) ¿Cuántos dispositivos de I/O pueden conectarse a un computador x86? **(1 pto.)**
Solución: Se pueden conectar máximo 15 dispositivos, debido al esquema esclavo-maestro utilizado por los PICs.

f) En un computador x86, describa por cuáles elementos del hardware pasa una serie de datos transmitidos desde la memoria RAM a la tarjeta de video mediante DMA. **(1 pto.)**
Solución: Los datos deben pasar primero a través del bus de memoria para llegar al North-Bridge. Luego viajan a través del bus gráfico hasta la memoria de la tarjeta de video.

Pregunta 2

a)
- En un computador el tiempo promedio de acceso a los datos es de 5ns. ¿Cuál es el hit-time de la caché, si su hit-rate es 0.99 y el tiempo de acceso a la memoria RAM es 400 veces mayor al hit-time de la caché? **(0.5 ptos.)**
  Solución:

$$T_{prom} = HT_{caché} + MR_{caché} \times MP$$
$$5ns = x + 0{,}01 \times 400x$$
$$x = \frac{5ns}{1 + 0{,}01 \times 400}$$
$$x = 1ns$$

- En un computador con memoria caché de 16KB, bloques de 8 palabras y tiempo de acceso de 10ns, se transfiere desde la memoria principal un bloque completo en 120ns, ¿cuál es el hit-rate de la caché para obtener un tiempo de acceso promedio de 20ns? **(0.5 ptos.)**
  Solución:

$$T_{prom} = HT_{caché} + MR_{caché} \times MP$$
$$20ns = 10ns + (1 - x) \times 120ns$$
$$x = 1 - \frac{10ns}{120ns}$$
$$x \approx 0{,}92$$

b) Un computador posee una memoria principal de 32KB con palabras de 16 bits y una memoria caché de 8KB dividida en conjuntos de 4 bloques y 64 palabras por bloque. Asuma que la caché está inicialmente vacía y el procesador solicita las siguientes direcciones de memoria de manera secuencial: 1, 2, ..., 4072. Este proceso se repita en total 10 veces. Si la caché es 10 veces más rápida que la memoria principal y asumiendo una política de reemplazo LRU, estime la mejora en tiempo de acceso dada por el uso de la caché. **(2 ptos.)**
Solución: En este problema, dado que los accesos a memoria son secuenciales y son menos que el tamaño de la caché, nunca se deberán hacer sustituciones de bloques, al ser un esquema N-way. De esta manera, a partir de la primera lectura de un bloque, todo el resto serán hits. Así, si asumimos que el tiempo de acceso a la caché es t y a la memoria principal es $10t$, tenemos que el tiempo total que demora un computador sin memoria caché en este proceso es: $4072 \times 10 \times 10t = 407200t$. Luego, dado que para las 4072 peticiones de memoria se utilizan en total $\lceil \frac{4072}{64} \rceil = 64$ bloques, el tiempo total de acceso para un sistema con caché será: $64 \times (10+1)t + (4072 - 64) \times t + 9 \times 4072t = 41360t$. Luego, el esquema que utiliza caché tomará $\frac{41360}{407200} \approx 0{,}1016$ del tiempo que tomaría sin caché.

c) La siguiente figura presenta el estado de la memoria principal de un computador con memoria virtual en un instante dado: En base a esto, asuma la siguiente situación

- Tamaño de cada página y de cada tabla de páginas es de 4 palabras.
- Existen dos procesos en ejecución, P1 y P2.
- Las páginas no existentes (no asociadas a marcos) se denotan con -.

Tabla de págs. P1

Dirección	Contenido
0	6
1	15
2	3
3	−2
4	−45
5	8
6	28
7	−2
8	9
9	−3
10	8
11	12

Tabla de págs. P2

12	0
13	−12
14	24
15	–
16	0
17	−12
18	−16
19	–
20	0
21	0
22	0
23	0

24	5
25	−3
26	−3
27	2
28	0
29	1
30	2
31	3
⋮	⋮

- En una tabla de páginas, el bit más significativo de cada palabra indica si la página está en memoria (0) o disco (1).
- Ambos procesos solicitan las siguientes direcciones virtuales: 0, 1, 4, 5, 8, 10, 12, 15.

Para cada proceso, transforme las direcciones virtuales en físicas. Si la transformación fue exitosa, indique el dato obtenido. En caso contrario, indique el tipo de page fault generado. **(3 ptos.)**
Solución:

Proc.	Dir. Virt.	Pág.	Offset	Marco	Valor
1	0	0	0	Disco	Page fault por marco en disco
1	1	0	1	Disco	Page fault por marco en disco
1	4	1	0	8	4
1	5	1	1	8	5
1	8	2	0	28	84
1	10	2	2	28	86
1	12	3	0	Disco	Page fault por marco en disco
1	15	3	3	Disco	Page fault por marco en disco
2	0	0	0	0	6
2	1	0	1	0	15
2	4	1	0	Disco	Page fault por marco en disco
2	5	1	1	Disco	Page fault por marco en disco
2	8	2	0	24	68
2	10	2	2	24	70
2	12	3	0	-	Page fault por marco no asignado
2	15	3	3	-	Page fault por marco no asignado

Pregunta 3

Un robot simple, conectado a un computador, es accesible mediante mapeo a memoria. Este robot se mueve en un espacio cuadrado infinito, en el cual cada grilla puede estar vacía o contener una muralla. El robot tiene comandos para ser prendido, apagado, avanzar 1 espacio hacia adelante, girar a la izquierda en 90^o y examinar lo que hay adelante. Cada vez que el robot encuentra desocupado, i.e., ha sido recién iniciado o ha terminado una acción, genera una interrupción para informar que es posible darle un nuevo comando.

a) Describa el mapa de memoria necesario para manejar el robot. **(1 pto.)**

Dirección	Contenido/Función asociada
0	Dirección ISR de manejo del robot.
1	Registro de comandos del robot.
2	Registro de estado del robot.
3-...	Memoria de uso libre

b) Defina el formato de los datos que recibirá el robot como comandos y que entregará este para informar su estado. **(1 pto.)**

Ubicación	Comando/Estado	Valor
Reg. Comandos	Encender	255
Reg. Comandos	Apagar	0
Reg. Comandos	Avanzar	1
Reg. Comandos	Girar Izq.	2
Reg. Comandos	Examinar	4
Reg. Estado	Recién encendido	0
Reg. Estado	Nada que informar	255
Reg. Estado	Espacio libre adelante	1
Reg. Estado	Muralla adelante	2

c) Escriba en assembly x86 la ISR asociada al control del robot, siguiendo el siguiente comportamiento: el robot avanza hasta encontrar una muralla, en cuyo caso girará a la izquierda hasta encontrar un espacio vacío para avanzar, teniendo la precaución de que el robot no retroceda. Asuma que el espacio ha sido diseñado para que el robot no se quede pegado girando eternamente. **(4 ptos.)**

Solución:

```
ISR_robot:
MOV BX, 0x0002
CMP [BX], 0x00
JE inicializar
CMP [BX], 0xFF
JE examinar
CMP [BX], 0x01
JE check_retroceso
JMP girar_izq

inicializar:
MOV BX, 0x0003  ;inicializamos variable en
MOV [BX], 0x00  ;direccion de memoria 3

examinar:
MOV BX, 0x0001
MOV [BX], 0x04
JMP end_isr

check_retroceso:
MOV BX, 0x0003
CMP [BX], 0x02  ;verificamos direccion de retroceso
JE girar_izq

avanzar:
MOV [BX], 0x00
MOV BX, 0x0001
MOV [BX], 0x01
JMP end_isr

girar:
MOV BX, 0x0003
ADD [BX], 0x01
MOV BX, 0x0001
MOV [BX], 0x02

end_isr:
IRET
```