

Cognoms: Nom:

3er Control Arquitectura de Computadors

Curs 2012-2013 Q2

Problema 1. (2,5 puntos)

Dado el siguiente código escrito en C:

<pre>typedef struct { char c1; char c2; } X; void Subr1 (char a[3], char *b, char c, X d) { char local1[3]; char *local2; char local3; X local4; ... }</pre>	<pre>void Subr2 (int *par1, int *par2) { int i=3; if (*par2!=7) Subr2(&i, par2); *par1 = i + *par2; }</pre>
---	---

- a) **Dibuja** el bloque de activación de la rutina Subr1, indicando claramente los desplazamientos respecto a ebp, el tamaño de todos los campos y las posiciones de todos los elementos de los vectores y de las estructuras.

- b) **Completa** en ensamblador del x86 el código de la rutina Subr2. El número entre paréntesis indica la cantidad máxima de instrucciones ensamblador que podéis utilizar en cada sección de código. Alguna sección podría no necesitar ninguna instrucción o menos del máximo indicado.

Subr2:		PUSH %EBP MOVL %ESP, %EBP
(2)		<div style="border: 1px solid black; height: 20px; width: 100%;"></div> <div style="border: 1px solid black; height: 20px; width: 100%;"></div>
		MOVL 12(%EBP), %EAX
(1)		<div style="border: 1px solid black; height: 20px; width: 100%;"></div>
		JE FIF PUSHL %EAX
(3)		<div style="border: 1px solid black; height: 20px; width: 100%;"></div> <div style="border: 1px solid black; height: 20px; width: 100%;"></div> <div style="border: 1px solid black; height: 20px; width: 100%;"></div>
		CALL SUBR2 ADD \$8, %ESP
(1)		<div style="border: 1px solid black; height: 20px; width: 100%;"></div>
FIF:		
(6)		<div style="border: 1px solid black; height: 20px; width: 100%;"></div> <div style="border: 1px solid black; height: 20px; width: 100%;"></div> <div style="border: 1px solid black; height: 20px; width: 100%;"></div> <div style="border: 1px solid black; height: 20px; width: 100%;"></div> <div style="border: 1px solid black; height: 20px; width: 100%;"></div> <div style="border: 1px solid black; height: 20px; width: 100%;"></div>
		MOVL %EBP, %ESP POPL %EBP RET

- c) Dado el número decimal 35,125, exprésalo en el formato IEEE 754 de simple precisión:

--	--	--

Cognoms: Nom:

3er control d'Arquitectura de Computadors

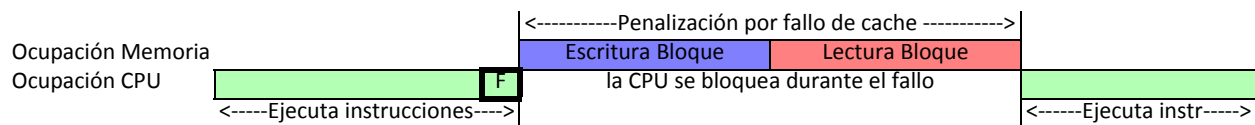
Curs 2012-2013 Q2

Problema 2. (3,5 puntos)

Se ha simulado la ejecución de un programa P en un procesador que denominaremos IDEAL. En este procesador IDEAL no hay ninguna penalización por fallo de cache. De esta simulación se ha obtenido que el programa P se ha ejecutado en 6×10^9 ciclos durante los que ha ejecutado 2×10^9 instrucciones, de las que 600×10^6 son instrucciones de acceso a datos (Load/Store) y se han producido 60×10^6 fallos en la cache de datos (los fallos en la cache de instrucciones son negligibles, con lo que los ignoraremos durante todo el problema). Suponemos que la probabilidad de fallar en cualquier ciclo es la misma y es independiente de que se haya fallado o no en el ciclo anterior.

- a) **Calcula** el CPI de P en el procesador IDEAL (CPI_{IDEAL}). **Calcula** el número medio de ciclos transcurridos entre 2 fallos.

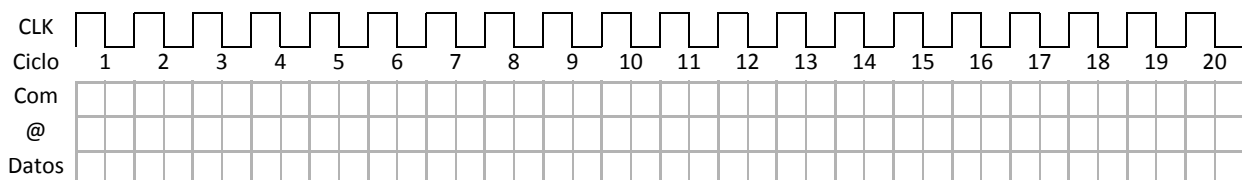
El mismo programa lo ejecutamos en un procesador real con las mismas características que el IDEAL, con la única diferencia que en caso de fallo en la cache de datos se bloquea la ejecución de instrucciones durante un cierto número de ciclos. La cache de datos sigue una política de escritura COPY BACK + WRITE ALLOCATE. En caso de fallo, si la línea reemplazada ha sido modificada, la penalización es de 90 ciclos, mientras que si no lo ha sido es sólo de 45 ciclos. Se sabe que durante la ejecución de P, en media 1/3 de los bloques de cache han sido modificados (dirty bit = 1). Este procesador funciona a una frecuencia de 2,4 GHz y lo denominamos CB+WA. La siguiente figura muestra el cronograma de un fallo en que la línea reemplazada ha sido modificada.



- b) **Calcula** el tiempo de ejecución de P en el procesador CB+WA

Este procesador está conectado a una memoria DDR3 SDRAM formada por un DIMM de 8 bytes de ancho con una latencia de fila de 5 ciclos, una latencia de columna de 4 ciclos y un tiempo para el comando PRECHARGE de 2 ciclos. Sabemos que el tamaño de bloque de cache es de 64 bytes.

- c) **Rellena** el siguiente cronograma indicando la ocupación de los distintos recursos de la SDRAM durante la lectura de un bloque de 64 bytes.



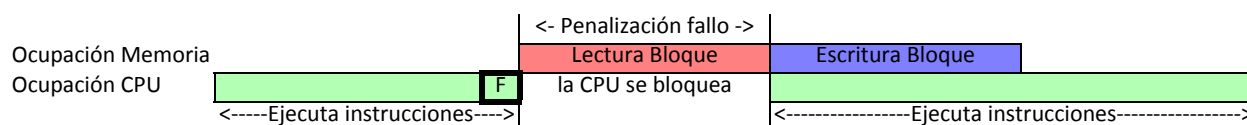
Sabemos que durante toda la duración de un acceso, el módulo DIMM disipa 0,6 W. Además durante la transmisión de los datos se disipan 0,75 W adicionales. La frecuencia del reloj de la DDR3 SDRAM es de 800 MHz.

- d) **Calcula** la potencia media y la energía consumida disipada por el DIMM durante la lectura de un bloque de 64 bytes.

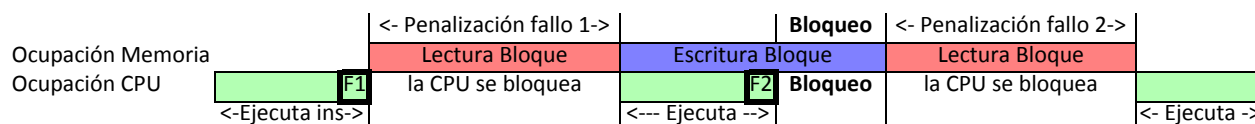
Sabemos que la escritura de un bloque de cache en la SDRAM consume la misma energía que una lectura.

- e) **Calcula** la energía total y la potencia media disipada por el DIMM durante la ejecución de P debida a la actividad causada por los fallos de cache.

Una posible mejora consiste en incorporar un buffer para almacenar el bloque reemplazado, de forma que podamos leer primero el bloque que ha provocado el fallo y a continuación escribir en memoria el bloque reemplazado. Esta implementación permite que el procesador pueda seguir ejecutando instrucciones y la cache pueda ser accedida durante la escritura del bloque reemplazado a memoria, tal como muestra la siguiente figura.



Disponemos de un buffer de una sola entrada que nos permite tener como máximo una escritura pendiente. Si durante la escritura del bloque causada por un fallo (F1) se produce un segundo fallo (F2), hay que esperar a que la jerarquía de memoria complete la escritura del bloque antes de que pueda empezar a servir el siguiente fallo, con lo que el procesador se bloqueará por unos ciclos adicionales (**Bloqueo**), tal como muestra la siguiente figura (en el ejemplo suponemos que el siguiente fallo reemplaza un bloque no modificado).



Durante la fase en que la CPU ejecuta instrucciones, éstas se ejecutan con la misma distribución que en el procesador IDEAL, por lo que el número medio de ciclos en que la CPU ejecuta instrucciones (sin contar el bloqueo debido a la lectura del bloque solicitado por el fallo F1) entre F1 y F2 será el mismo.

- f) **Calcula** la probabilidad de que se produzca un segundo fallo durante la escritura de un bloque reemplazado

Hemos medido que, si se produce un segundo fallo durante el intervalo de escritura de un bloque anterior, en media la CPU se bloquea durante 22 ciclos (ciclos correspondientes al intervalo **Bloqueo** de la figura anterior) antes de que se pueda iniciar la lectura del bloque que ha causado el segundo fallo.

- g) **Calcula** el numero de ciclos necesario para ejecutar P en el procesador con dicha mejora

Cognoms: Nom:

3er Control Arquitectura de Computadors

Curs 2012-2013 Q2

Problema 3. (4 puntos)

- a) **Describe** el funcionamiento de un RAID 6 de 6 discos. En la descripción no puede faltar un esquema de cómo se distribuyen los datos, porcentaje de información redundante, número de discos que han de fallar para que el RAID 6 deje de ser operativo, etc.

Disponemos de discos físicos de 3 TB de capacidad por disco, que ofrecen un ancho de banda efectivo de 250 Mbytes/s por disco. Con estos discos queremos montar el RAID 6 con 6 discos del apartado a).

- b) En caso de que falle un disco del RAID 6 del apartado anterior, **indica** de cuántos discos hay que leer para recuperar la información y **describe** claramente la forma en que la información se recupera.

Disponemos de discos físicos de 3 TB de capacidad por disco que ofrecen un ancho de banda efectivo de 250 Mbytes/s por disco. Con estos discos queremos montar un esquema RAID 61 con una capacidad de almacenamiento ÚTIL de 15 TB de datos usando el mínimo número de discos.

- c) **Dibuja** este esquema e **indica** el ancho de banda efectivo (datos útiles) máximo de lectura del RAID 61.

- d) **Indica** el ancho de banda efectivo máximo de escrituras que puede conseguirse en el esquema RAID 61 del apartado c) y **justifica** tu respuesta.

Cada uno de los RAIDs 6 del apartado c) tiene una fuente de alimentación con un MTTF de 200.000 horas y un MTTR de 15 horas. Suponiendo que los discos nunca fallarán y teniendo en cuenta exclusivamente las fuentes de alimentación.

- e) **Calcula** el MTTF del RAID 61.

Disponemos de una aplicación A que tiene dos fases de ejecución (lectura y cálculo) y se ejecuta en un solo procesador P con un único disco duro D en un tiempo T. La fase de lectura supone el 30% del tiempo de ejecución y el 80% de la fase de cálculo es paralelizable. Ejecutamos la aplicación en un sistema multiprocesador MP que tiene 4 procesadores P y un RAID cuyo ancho de banda efectivo de lectura es 10 veces el del disco duro D. Suponiendo que la paralelización se hace de forma perfecta.

- f) **Calcula** el speedup al ejecutar la aplicación A en el sistema MP.