Cognoms: . . . . Nom: . . . .

2on control d'Arquitectura de Computadors **Curs 2012-2013 Q2**

Problema 1. (3 puntos)

Dado el siguiente código escrito en C:

```
void Subr1 (double *a, int b[3], char c, short d) {
  double *local1;
  int local2[3];
  char local3;
  short local4;
  ...
}

int Subr3 (int *a, int *b);

int Subr2 (int *par) {
  int i;
  if (*par!=7)
     Subr3(&i, par);
  return i+*par;
}
```

a) Dibuja el bloque de activación de la rutina Subr1, indicando claramente los desplazamientos respecto a ebp y el tamaño de todos los campos.

b) Traduce a ensamblador del x86 la rutina Subr2.

Cognoms: . . . . . Nom: . . . . . .

2on control d'Arquitectura de Computadors **Curs 2012-2013 Q2**

Problema 2. (3 puntos)

a) **Define** de forma clara y concisa el comportamiento una cache con política write through + write NO allocate en caso de escritura de un byte con acierto y en caso de escritura de un byte con fallo:

b) Disponemos de un procesador de 16 bits con direcciones de 20 bits que tiene una memoria cache de datos con las siguientes características:

- 3-asociativa, con algoritmo de reemplazo LRU
- 256 bytes por bloque
- 12 bloques en la cache
- política de escritura: copy back + write allocate

El contenido inicial de la cache (por simplicidad hemos utilizado el número de bloque de MP, en vez del tag) es el siguiente:

conjunto 0	DB	conjunto 1	DB	conjunto 2	DB	conjunto 3	DB
984	0	441	0	666	0	0A7	1
98C	0	A19	0	0A6	0	45B	0
000	1	-	-	002	0	F2F	0

DB es el dirty bit. La información de reemplazo está implícita en la posición. Las posiciones inferiores corresponden a los bloques que llevan más tiempo sin utilizarse. Las posiciones superiores corresponden a los últimos bloques utilizados. Por ejemplo, en el conjunto 3, el bloque 0A7 es el último utilizado, y el bloque F2F el que lleva más tiempo sin ser utilizado.

Rellenad la siguiente tabla, indicando para cada referencia, el número de bloque de MP que le corresponde, a qué conjunto de MC va a parar, si es acierto o fallo, si hay lectura de MP, si hay escritura en MP y el bloque reemplazado cuando proceda.

dirección (hex)	bloque MP	conjunto MC	¿acierto o fallo?	lectura MP ¿si/no?	Escritura MP ¿si/no?	¿bloque reemplazado?
lect 00441						
esc 0A666						
escr 00002						
lect 98402						

¿Cuál es el contenido final de la Memoria Cache?

conjunto 0	DB	conjunto 1	DB	conjunto 2	DB	conjunto 3	DB

Dado el siguiente código escrito en ensamblador del IA32:

```
        movl $0, %ebx
        movl $0, %esi
for:    cmpl $256*1000, %esi
        jge end
(a)     movl (%ebx, %esi, 4), %eax
(b)     addl %eax, 4*1024(%ebx, %esi, 4)
(c)     movl %eax, 12*1024(%ebx, %esi, 4)
        addl $256, %esi
        jmp for
end:
```

c) Calcula la cantidad de aciertos y fallos de cache para el acceso marcado con la etiqueta (b) suponiendo una cache completamente asociativa de 8 Kbytes y líneas de 1Kbyte.

Suponiendo que la memoria utiliza **páginas de tamaño 4KB** y que utilizamos un **TLB de 4 entradas (reemplazo LRU)**, responde a las siguientes preguntas:

d) Para cada uno de los accesos (etiquetas a, b, c), indica a qué página de la memoria virtual se accede en cada una de las 17 primeras iteraciones.

	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
a																	
b																	
c																	

e) Calcula la cantidad de **aciertos de TLB**, en todo el bucle: . . . . . .

f) Calcula la cantidad de **fallos de TLB**, en todo el bucle: . . . . . .

Cognoms: . . . . . Nom: . . . . . .

2on control d'Arquitectura de Computadors **Curs 2012-2013 Q2**

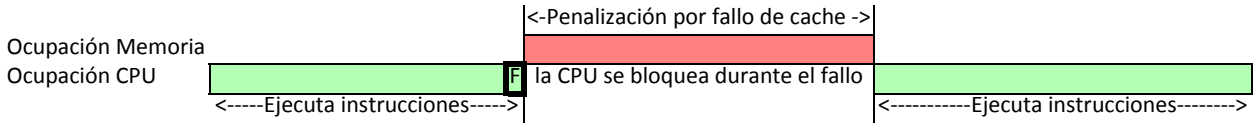
Problema 3. (4 puntos)

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 $5x10^9$ ciclos durante los que ha ejecutado $2x10^9$ instrucciones, de las que $500x10^6$ son instrucciones de acceso a datos (Load/Store) y se han producido $50x10^6$ 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) **Calculad** el CPI de P en el procesador IDEAL (CPI_{IDEAL}).

b) **Calculad** 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 numero de ciclos que corresponden al tiempo medio de penalización por fallo de cache (Tpf) hasta que se resuelve el fallo. La siguiente figura ilustra este hecho cuando se detecta un fallo (F).

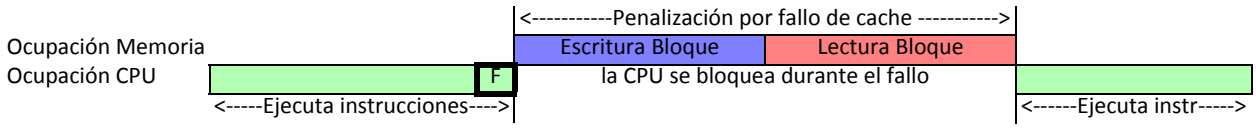


El programa P se ha ejecutado en el procesador real (que llamaremos procesador PA) en 4 segundos. Este procesador PA funciona a una frecuencia de 2 GHz.

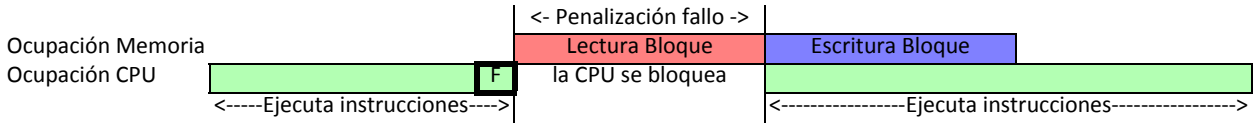
c) **Calculad** el CPI de P en el procesador PA (CPI_{PA})

d) **Calculad** el número medio de ciclos de penalización por fallo de cache (Tpf)

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). La siguiente figura muestra el cronograma de un fallo en que la línea reemplazada ha sido modificada.



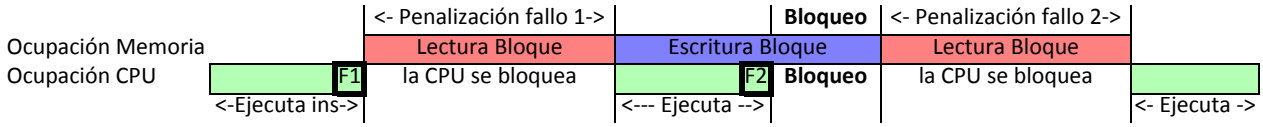
Una posible mejora (vista en clase de teoría) consiste en incorporar un buffer para almacenar el bloque reemplazado, de forma que podamos leer primero el bloque que ha provocado el fallo y a continuación escribir en memoria el bloque reemplazado. Esta implementación permite que el procesador pueda seguir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



De momento supondremos la situación ideal (aunque no realista) en que nunca se produce un fallo de cache mientras se realiza la escritura del bloque reemplazado. A este procesador lo denominaremos procesador PB

e) **Calculad** el numero de ciclos necesario para ejecutar P en el procesador PB

En la implementación real (que denominamos procesador PC) dispondremos 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 estas 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) **Calculad** 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) **Calculad** el numero de ciclos necesario para ejecutar P en el procesador PC

h) Calcula el % de speed-up del proceasdor PC respecto el PA