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

3er Control Arquitectura de Computadors **Curs 2012-2013 Q2**

Problema 1. (2,5 puntos)

Dado el siguiente código escrito en C:

```
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;
  ...
}
```

```
void Subr2 (int *par1, int *par2) {
  int i=3;
  if (*par2!=7)
     Subr2(&i, par2);
  *par1 = i + *par2;
}
```

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.A.

1	local1[0]	-12
1	local1[1]	-11
1	local1[2]	-10
1	---	-9
4	local2	-8
1	local3	-4
1	local4.c1	-3
1	local4.c2	-2
1	---	-1
4	ebp viejo	<- %ebp
4	dir. retorno	
4	&a	8
4	b	12
1	c	16
3	---	
1	d.c1	20
1	d.c2	21
2	---	

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)
        SUBL $4, %ESP
        MOVL $3, -4(%EBP)
        MOVL 12(%EBP), %EAX
(1)
        CMPL $7, (%EAX)
        JE FIF
        PUSHL %EAX
(3)
        LEAL -4(%EBP), %EAX
        PUSHL %EAX
        
        CALL SUBR2
        ADD $8, %ESP
(1)
        
FIF:
(6)
        MOVL 12(%EBP), %EAX
        MOVL (%EAX), %EAX
        ADDL -4(%EBP), %EAX
        MOVL 8(%EBP), %ECX
        MOVL %EAX, (%ECX)
        
        MOVL %EBP, %ESP
        POPL %EBP
        RET
```

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

0	10000100	**00011001000000000000000**

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

3er control d'Arquitectura de Computadors **Curs 2012-2013 Q2**

Problema 2. (3,5 puntos)

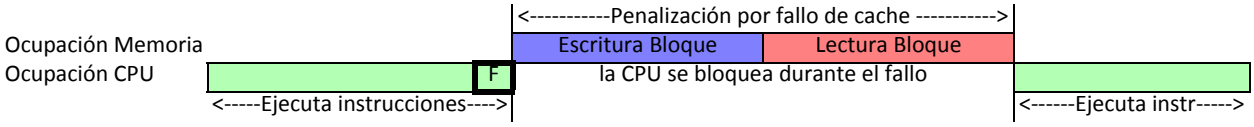
Se ha simulado la ejecución de un programa P en un procesador que denominaremos IDEAL. En este procesador IDEAL no hay ninguna penalización por fallo de cache. De esta simulación se ha obtenido que el programa P se ha ejecutado en $6x10^9$ ciclos durante los que ha ejecutado $2x10^9$ instrucciones, de las que $600x10^6$ son instrucciones de acceso a datos (Load/Store) y se han producido $60x10^6$ 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.

CPI_{IDEAL} = $6x10^9$ ciclos / $2x10^9$ instrucciones = 3 ciclos / instrucción

$6x10^9$ ciclos / $60x10^6$ fallos = 100 ciclos entre 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. 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.

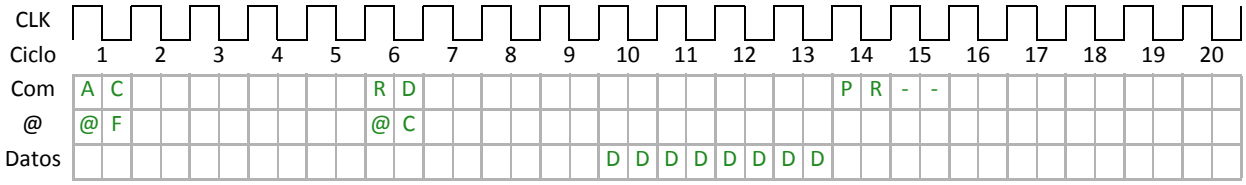


b) **Calcula** el tiempo de ejecución de P en el procesador CB+WA

$ciclos_{CB+WA}$ = $ciclos_{IDEAL}$ + #fallos(D=0) * Penalización(D=0) + #fallos(D=1) * Penalización(D=1)
$ciclos_{CB+WA}$ = $6x10^9$ ciclos + 2/3*$60x10^6$ fallos * 45 ciclos/fallo + 1/3*$60x10^6$ fallos * 90 ciclos/fallo= $9,6x10^9$ ciclos
Texe = $9,6x10^9$ ciclos/$2,4x10^9$ ciclos/s = 4 s

Este procesador esta connectado 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 modulo 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 energia consumida disipada por el DIMM durante la lectura de un bloque de 64 bytes.

Pmedia = (15 ciclos *0,6 W + 4 ciclos * 0,7,5 W) / 15 ciclos = 0,8 W

Energia = 0,8W *15 ciclos / $0{,}8 \times 10^9$ ciclos/s = 15 nJ

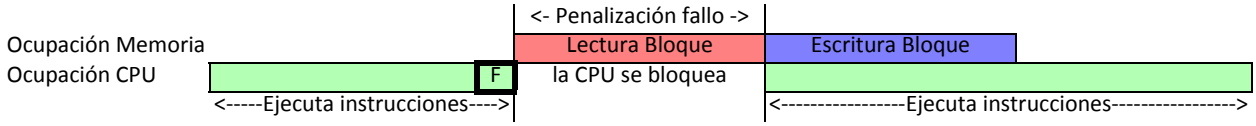
Sabemos que la escritura de un bloque de cache en la SDRAM consume la misma energia que una lectura.

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

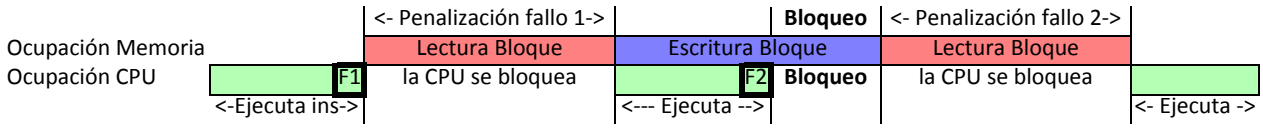
Energia = 15 nJ/acceso * (60×10^6 lecturas + 20×10^6 escrituras) = 1,2 Joule

Potencia = 1,2 Joule/ 4 segundos = 0,3 W

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

probabilidad de fallar en un ciclo es p = 1/100 (inversa del tiempo medio entre fallos)
probabilidad de tener un fallo en 45 ciclos (intervalo de servicio de F1) es 1 - probabilidad de no fallar en ningún ciclo (repetimos un proceso independiente 45 veces con probabilidad p)

P(fallo en el intervalo) = $1 - (1 - p)^{45} = 1-(1-1/100)^{45} = 0{,}364$

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

penalización por bloqueo = 22 ciclos
$\text{ciclos}_{\text{mejora}} = \text{ciclos}_{\text{IDEAL}}$ + #fallos*Penal Lect + #fallos * bloques mod *prob bloqueo * Penal por bloqueo =
= 6×10^9 ciclos + 60×10^6 fallos * 45 ciclos/fallo + 60×10^6 fallos * 1/3 * 0,364 * 22 ciclos/fallo= $8{,}86 \times 10^9$ ciclos

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, numero de discos que han de fallar para que el RAID 6 deje de ser operativo, etc.

Dibujo.

En RAID 6 la información redundante se guarda en forma de paridad (P) y mediante una función basada en códigos Reed Solomon (Q). Por tanto, se requieren 2 discos para guardar información redundante del resto. En un RAID 6 de 6 discos, 2 serían para guardar información redundante y 4 de datos, por lo que el porcentaje de información redundante es de 1/3.

La paridad P + Q permite soportar el fallo de hasta 2 discos.

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.

Para recuperar la información de 1 disco es necesario leer de todos los discos que tengan datos, y usar el disco de paridad. Por lo tanto, es preciso de leer de los otros tres discos de datos + el disco que contiene la paridad correspondiente (lectura de 4 discos).

Para recuperar la información, se lee la tira correspondiente en los 4 discos indicados, se realiza la XOR bit a bit y la tira resultado se envía al destino o se escribe en el nuevo disco, en caso de que el disco se esté reconstruyendo.

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.

Se necesitan 5 discos de datos, + 2 de redundancia P y Q, para implementar el nivel 6 del RAID 61.

Se necesita replicar el esquema anterior para implementar el nivel 1 del RAID 61. Por tanto, hacen falta 14 discos.

Ancho de banda efectivo: Se puede leer de cualquiera de los dos grupos de RAID6, y dentro de cada uno se puede acceder a todos los discos si hay suficicntes peticiones, por lo tanto el ancho de banda efectivo máximo:

AB max = 250 MB/s*disco x 14 discos = 3500 MB/s

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.

Para realizar una escritura se necesitan 3 lecturas y 3 escrituras en cada uno de los dos grupos de 6 discos del RAID. Por lo tanto, una escritura requiere 12 accesos a disco.

Si hay suficientes escrituras a realizar, el ancho de banda máximo es:

AB max = 14 discos x 1 escritura*disco/12 discos x 250 MB/s*disco = 291,66 MB/s

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.

Para que el sistema falle tienen que fallar las dos fuentes de alimentación, ya que si falla una de las dos se puede servir la información desde el otro grupo de 7 discos del RAID. Por lo tanto (ver transparencia 39 del tema "Fundamentos":

MTTF= 200000^2 / (2*15) = 1.333.333.333,33 horas = 55.555.555,55 días = 152.207 años

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.

T = 0,3 T + 0,7 (0,8 T + 0,2 T).

La fase que dura 0,3 T se reduce en 1/10 y la parte que dura 0,8 T se reduce 1/4. Por lo tanto:

T' = 0,03 T + 0,7 (0,2 T + 0,2 T) = 0,03 T + 0.14 T + 0,14 T = 0,31 T

Speedup = T/T' = 1/0,31 = 3,226