

Cognoms: Nom:

Examen Final d'Arquitectura de Computadors

Curs 2012-2013 Q2

Problema 1. (1,5 puntos)

a) **Define** de forma clara y concisa el comportamiento de una cache con política copy back + write allocate :

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:

```

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

b) 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																	

c) **Calcula** la cantidad de **aciertos y fallos de TLB**, en todo el bucle:

Problema 2. (2 puntos)

Dado el siguiente código escrito en C:

```
typedef struct {  
    char a;  
    int b;  
    short c;  
    char d;  
    int e[4];  
} s1;  
  
typedef struct {  
    s1 f[100];  
    int d;  
} s2;  
  
int examen(s1 j, s2 *k, char m[10], short n){  
    short u;  
    char v[3];  
    int w;  
    ...  
}
```

b) **Dibuja** el bloque de activación de la función examen, indicando claramente los desplazamientos relativos al registro EBP necesarios para acceder a los parámetros y a las variables locales.

c) **Escribe** la expresión aritmética que permite calcular la dirección del elemento $x.f[y].e[i].d$, siendo x una variable de tipo s2 e y una variable de tipo s1:

d) **Traduce** la siguiente sentencia a ensamblador del x86, suponiendo que está dentro de la función examen. Se valorará la optimización en el código. Escribe claramente la expresión aritmética a traducir.

```
j.e[w]=0;
```

Cognoms: Nom:

Examen Final d'Arquitectura de Computadors

Curs 2012-2013 Q2

Problema 3. (3 puntos)

Tenemos un procesador (que llamaremos procesador original) que funciona a una frecuencia de 4 GHz. Este procesador no tiene ningún tipo de jerarquía de memoria, todos los accesos se realizan directamente sobre memoria principal. En este procesador hemos ejecutado una aplicación A (que usaremos a lo largo de todo el problema) y hemos obtenido los siguientes datos: tiempo de ejecución 16 s, 2×10^9 instrucciones ejecutadas, 3×10^9 accesos a memoria. Sabemos además que cada acceso a memoria tarda 21 ciclos.

- a) **Calcula** el CPI del procesador original al ejecutar la aplicación A (a este CPI lo llamaremos $CPI_{original}$).

- b) **Calcula** el CPI de un procesador (que llamaremos procesador ideal) en que cada acceso tardase 1 ciclo (a este CPI lo denominaremos CPI_{ideal}).

Para mejorar el rendimiento respecto al procesador original, añadimos una cache 2-asociativa al procesador (que denominaremos procesador 2A). Para poder mantener la frecuencia de 4 GHz, los accesos a la cache 2-asociativa tardan 2 ciclos, es decir que en caso de acierto tenemos una penalización de 1 ciclo respecto al procesador ideal. La penalización media en caso de fallo es de 25 ciclos (también respecto al procesador ideal). La aplicación A ejecutada en el procesador 2A tarda 3,55 segundos.

- c) **Calcula** la tasa de fallos de la cache 2-asociativa.

Uno de los ingenieros ha sugerido el uso de un predictor de vía.

- d) **Explica** cómo funciona una cache asociativa por conjuntos con predictor de vía y sus ventajas e inconvenientes respecto una cache igual pero sin predictor de vía.

El predictor sugerido tendría un impacto negligible tanto en área como en la frecuencia y consumo del procesador. Al ejecutar la aplicación A, el predictor de vía tiene una tasa de aciertos del 80%. Al procesador con cache 2 asociativa y predictor de vía lo denominaremos procesador P. En caso de que el predictor acierte la vía no hay penalización respecto al procesador ideal, si hay fallo de predictor pero acierto de cache se incurre en un ciclo de penalización (tal como ocurría con la cache 2 asociativa), y finalmente, si es fallo de predictor y también de cache, la penalización es de 25 ciclos (también la misma que la cache 2 asociativa). Además del predictor, se ha aprovechado para mejorar el algoritmo de reemplazo de la cache, por lo que la tasa de fallos de la misma es de solo un 5%.

- e) **Calcula** el tiempo de ejecución de la aplicación A en el procesador P.

Queremos saber la potencia media debida a conmutación de la jerarquía de memoria del procesador P. Ignoraremos por tanto la potencia disipada por fugas así como la potencia de conmutación del procesador, y también la del predictor, que ya hemos comentado que tiene un impacto despreciable. Sabemos que cada vez que se accede una vía de la cache se consumen 5 nJ (nanojoules) y cada vez que hay un fallo de cache se consumen 44 nJ adicionales.

- f) **Calcula** la potencia media debida a conmutación de la jerarquía de memoria del procesador P al ejecutar la aplicación A

Cognoms: Nom:

Examen Final d'Arquitectura de Computadors

Curs 2012-2013 Q2

Problema 4. (3,5 puntos)

Se ha medido la ejecución de un programa en un sistema con un solo procesador y un solo disco (un PC de sobremesa) y se ha visto que su tiempo de ejecución es de 300 horas.

Dado que el coste en tiempo es muy alto, deseamos ejecutar el programa (que en parte es paralelizable) en un sistema multiprocesador con varios procesadores y discos idénticos al de nuestro PC. Para poder estimar el rendimiento del programa hemos medido (en el PC) que el programa se ejecuta en 3 fases bien diferenciadas:

Fase 1: Código SECUENCIAL que no puede paralelizarse, ocupa el 5% del tiempo en la ejecución en el PC.

Fase 2: Código TOTALMENTE PARALELIZABLE, ocupa el 85% del tiempo si se ejecuta el programa en el PC.

Fase 3: Código de E/S que pasa todo su tiempo ESCRIBIENDO en disco, ocupa el 10% del tiempo en la ejecución en el PC.

- a) Calcula la ganancia máxima en velocidad que se puede conseguir en este programa si dispusiéramos para ejecutarlo de un supercomputador compuesto por un número infinito de procesadores y discos.

En realidad, aunque la Fase 2 sea muy paralelizable, no es posible realizar una paralelización completa debido al tiempo de sincronización entre los distintos procesos. Se calcula que el tiempo de sincronización aumenta conforme aumenta el número de procesadores por lo que hay un límite a la cantidad de procesadores que podemos usar ganando velocidad. Hemos averiguado que por cada procesador que añadimos el tiempo de ejecución aumenta en 30 horas (con respecto a la paralelización ideal de la fase 2).

- b) Calcula cual es el número ideal de procesadores y la ganancia en porcentaje del programa cuando se ejecuta en un entorno con dicho número de procesadores y un disco.

Nuestro supercomputador, además de poseer muchos procesadores también dispone de un RAID de discos. Esto nos permite paralelizar la Fase 3 ya que en esta fase hay suficientes accesos como para saturar el ancho de banda de los discos. Sabemos además que todos los accesos son aleatorios.

- c) El RAID de discos del que disponemos tiene 30 discos y puede configurarse como RAID 0, 1(2 grupos de 15 discos), 5, 50 (5 grupos de 6 discos) o 6. Si solo nos interesara el rendimiento, ¿cual es la mejor opción de configuración? ¿Cual sería la respuesta si la fase 3 fueran todo lecturas? ¿Y si solo nos interesara la capacidad?

Nuestro sistema supercomputador está compuesto por un RAID 50 de 30 discos, cada uno de los cuales es igual al del PC de sobremesa. A partir de aquí suponemos que siempre que hablamos del supercomputador supondremos que estamos hablando del número ideal de procesadores y un RAID 50 de 30 discos.

- d) Calcula cuanto tiempo tardará nuestro programa ejecutado en el supercomputador.

Sabemos que el programa contiene 10^{17} instrucciones dinámicas de las cuales, en la fase dos, $864 * 10^{14}$ son instrucciones de coma flotante que implementan un total de $72 * 10^{14}$ operaciones de coma flotante en simple precisión.

- e) Calcula a cuantos MIPS y MFLOPs se ejecuta el programa en el PC de sobremesa y en el supercomputador con el número ideal de procesadores sabiendo que las instrucciones dinámicas de sincronización son $1 * 10^{14}$ por cada procesador extra.

- f) Comenta de forma razonada que sistema (el PC de sobremesa o el supercomputador) presenta una mejor eficiencia energética suponiendo que los procesadores y los discos son los elementos que disipan más del 99% de la energía en cualquiera de los dos sistemas.

Visto el pobre rendimiento obtenido y suponiendo que la totalidad del tiempo de la fase 2 se consume en las instrucciones dinámicas de coma flotante (el resto de fases no tienen dichas instrucciones) decidimos añadir al procesador del supercomputador una unidad SIMD de coma flotante de 256 bits que, para el código analizado, nos permite agrupar cada 8 instrucciones dinámicas de coma flotante en una que se ejecuta a la misma velocidad que las anteriores.

- g) Estimad de forma razonada cual sería la nueva paralelización ideal de la fase 2 del programa con los nuevos procesadores.