COGNOMS (en majúscules): . . . . . .

NOM (en majúscules): . . . . . .

Duració: 2 hores

Problema 1. (3 puntos)

Dado el siguiente código escrito en C:

```
int repetida(int a, int b[2][2], char c, char d) {
  int x[2],
  char y, z;
  . . .

  return (x[1]+x[0])
}
```

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

La rutina repetida supone el 30% del tiempo de ejecución de un programa. Nuestro objetivo es conseguir un speed-up del 25% en todo el programa y tenemos dos opciones para ello: (O1) Mejorar el rendimiento de la rutina repetida y (O2) mejorar el rendimiento del resto del programa.

b) **Calcula** la ganancia (speed-up) que necesitamos para la opción (O1) (mejorar la rutina).

c) **Calcula** la ganancia (speed-up) que necesitamos para la opción (O2) (mejorar el resto del programa).

Al medir el consumo de potencia nos damos cuenta de que la rutina repetida tiene un consumo medio de 6W mientras que el resto del programa tiene un consumo medio de 4W. Las optimizaciones realizadas no afectan al consumo medio.

d) **Determina** con cuál de las dos posibles optimizaciones (O1/O2) el programa consumiría menos energía. La justificación tiene que ser respaldada numéricamente.

COGNOMS (en majúscules): . . . . . .

NOM (en majúscules): . . . . . .

Duració: 2 hores

Problema 2. (3,5 puntos)

Un procesador con direcciones físicas de 32 bits tiene una cache de primer nivel (L1) 4-asociativa y con tamaño de bloque 64 bytes. Las etiquetas (TAGS) de la cache son de 18 bits.

a) **Calcula** el numero de bloques (líneas) de la cache.

El procesador dispone ademas de un TLB que se accede en paralelo a L1.

b) **Calcula** el tamaño mínimo que pueden tener las páginas de memoria virtual para que sea posible el acceso paralelo a cache y TLB.

En los siguientes apartados asumiremos que la influencia de los fallos de TLB y de los fallos de página es despreciable y estudiaremos exclusivamente el impacto de la cache. Para simplificar el problema también asumiremos que todos los accesos son lecturas.

Para un programa (P1) el tiempo medio de acceso a memoria (Tma) es de 3 ciclos. En caso de acierto en L1 el tiempo de acceso es de un ciclo, mientras que en caso de fallo hay una penalización (Tpf) de 80 ciclos adicionales.

c) **Calcula** la tasa de fallos de la cache para el programa P1.

Los diseñadores del procesador quieren mejorar el tiempo medio de acceso (Tma) de forma que la ganancia del nuevo Tma respecto al viejo sea del 50% (piensa bien que significa una ganancia del 50%). Para ello están considerando añadir un segundo nivel de cache (L2). En caso de acierto en L1 el tiempo de acceso sigue siendo de 1 ciclo. En caso de fallo en L1 y acierto en L2 el tiempo de penalización es de solo 16 ciclos. Finalmente, en caso de fallar también en L2, además de la penalización por acceder a L2 se incurre en una penalización adicional de 80 ciclos más para acceder a memoria.

d) **Calcula** la tasa de fallos local que debería tener la cache L2 para el programa P1.

Finalmente se ha determinado que el ahorro de energía es fundamental. Se ha optado por no incluir el segundo nivel de cache y se ha rediseñado la cache L1 para incluir un predictor de vía (PV). Recordemos que la L1 es 4-asociativa y cada vía (v) tiene su memoria de etiquetas (o tags) T(v) y su memoria de datos D(v). Un acceso a una memoria de etiquetas T(v) consume 5 nJ, mientras que un acceso a una memoria de datos D(v) consume 20 nJ. El consumo del predictor es despreciable. A continuación se describe el funcionamiento de la nueva cache con predictor de vía.

Un acceso a cache puede tardar entre 1 y 3 ciclos (además de la penalización correspondiente en caso de fallo de cache). Sea vp la vía predicha por el predictor PV de forma anticipada:

- Ciclo 1: Se accede a T(vp) (memoria de etiquetas de la vía) y a D(vp) (memoria de datos de la vía). En caso de que PV haya acertado la vía, la CPU obtiene el dato en este ciclo. En caso de fallo de predictor se procede al Ciclo 2.
- Ciclo 2: Se acede a la memoria de etiquetas de todas las vías T(0), T(1), T(2) y T(3)
  - Si es acierto de cache se determina la vía real en que se encuentra el dato (vr) y se procede al Ciclo 3.
  - Si es fallo de cache se incurre en una penalización adicional de 80 ciclos.
- Ciclo 3: Se accede a la memoria de datos de la vía real D(vr).

e) **Explica** brevemente cómo se determina si es acierto o fallo de predictor en el Ciclo 1

f) **Explica** brevemente las ventajas de usar un ciclo adicional (Ciclo 3) para acceder a los datos en caso de acierto de cache en vez de acceder en paralelo a los datos en el Ciclo 2

Un programa (P2) realiza 10^9 accesos a memoria. Queremos comparar la nueva organización con predictor, con una organización de cache con acceso paralelo a datos y etiquetas. Para P2 la cache (en ambos casos ya que es la misma) tiene una tasa de fallos del 5%. El predictor de vía PV tiene una tasa de aciertos del 85% respecto al total de accesos. Recordemos que en la cache paralela, en caso de acierto en L1 el tiempo de acceso es de un ciclo, mientras que en caso de fallo hay una penalización de 80 ciclos adicionales.

g) **Calcula** cuantos ciclos adicionales tarda el programa P2 con el predictor de vía respecto la cache paralela.

COGNOMS (en majúscules): . . . . . .

NOM (en majúscules): . . . . . .

Duració: 2 hores

Problema 3. (3,5 puntos)

Se quiere diseñar la memoria cache de datos de primer nivel para un procesador de tipo Load/Store. La política de escritura de la memoria es *Copy Back* y *Write Allocate*:

Se han obtenido por simulación las siguientes medidas para un determinado programa:

- porcentaje de escrituras (sobre el total de accesos): 10%
- porcentaje de bloques modificados: 40%
- tasa de aciertos: 0,9

En caso de fallo, el bloque de MP se escribe en la MC y posteriormente se realiza la operación de lectura/escritura en la MC. En el caso de reemplazar un bloque modificado (*dirty*), primero se escribe y después se lee. El tiempo de acceso (Tsa) a memoria cache (MC) es de 10 ns tanto para lectura como escritura. El tiempo de acceso a memoria principal (MP) para escribir una palabra es de 100 ns. Para leer o escribir un bloque en la MP se emplean 120 ns.

a) **Calcula** el tiempo empleado en realizar 1000 accesos consecutivos

Un computador tiene una memoria cache unificada (MC) conectada a una memoria principal (MP) DDR3 mediante un bus de 64 bits. Para enviar un bloque desde la MP a la MC la MP está ocupada durante 9 ciclos, desglosados de la siguiente forma: 2 ciclos de latencia de fila + 2 ciclos de latencia de columna + 4 ciclos para enviar todos los datos + 1 ciclo de precarga.

b) ¿**Cuál** es el tamaño de bloque de la cache?

Dado el siguiente código escrito en ensamblador del x86 (nótese que el escalado es 8 en los accesos a memoria):

```
      movl $0, %ebx
      movl $0, %esi
for:  cmpl $512*1000, %esi
      jge end
 (a)  movl (%ebx, %esi, 8), %eax          # escalado 8
 (b)  addl %eax, 2*4*1024(%ebx, %esi, 8)  # escalado 8
 (c)  movl %eax, 4*4*1024(%ebx, %esi, 8)  # escalado 8
      addl $1, %esi
      jmp for
end:
```

Sabemos que el código se ejecuta en un sistema con memoria cache y memoria virtual. La memoria virtual utiliza páginas de tamaño 4KB y disponemos de un TLB de 4 entradas y reemplazo LRU. La memoria cache de datos (únicos accesos a memoria que contemplaremos en este problema) es *Copy Back + Write Allocate*, completamente asociativa con reemplazo LRU, tamaño 8 KB y 16 bytes por bloque. Responde a las siguientes preguntas:

c) **Calcula** la cantidad de aciertos y de fallos de cache, en todo el código.

d) Para cada uno de los accesos indicados (etiquetas a, b, c), **indica** a qué página de la memoria virtual se accede en cada una de las siguientes iteraciones del bucle.

iteración	0	1*512	2*512	3*512	4*512	5*512	6*512	7*512	8*512	9*512
a										
b										
c										

e) **Calcula** la cantidad de aciertos y de fallos de TLB, en todo el código.