COGNOMS:														
NOM:								]						

**IMPORTANTE leer atentamente antes de empezar el examen:** Escriba los apellidos y el nombre antes de empezar el examen. Escriba un solo carácter por recuadro, en mayúsculas y lo más claramente posible. Es importante que no haya tachones ni borrones y que cada carácter quede enmarcado dentro de su recuadro sin llegar a tocar los bordes. Use un único cuadro en blanco para separar los apellidos y nombres compuestos si es el caso. No escriba fuera de los recuadros, todo lo que haya fuera de ellos es ignorado.

## Problema 1. (3,2 puntos)

Considerar el siguiente fragmento de código escrito en lenguaje C y correspondiente a la forma IJK del producto de matrices C = A \* B trabajado en la práctica 8:

```
main()
{
  float A[M][M], B[M][M], C[M][M];
  int i,j,k;
    . . .
  for (i = 0; i < M; i++)
    for (j = 0; j < M; j++)
    for (k = 0; k < M; k++)
    C[i][j] = C[i][j] + A[i][k] * B[k][j];
    . . .
}</pre>
```

Suponer que disponemos de una cache de datos de tamaño ilimitado, totalmente asociativa, con líneas de 16 bytes, y que las matrices A, B y C no se encuentran en cache al comienzo del código. El tamaño de página de nuestro sistema es de 16 Kbytes. La constante M toma valores grandes (M > 256).

Res	ponde a las siguientes preguntas:
<u>a)</u>	Indica si al acceder a las matrices A, B y C la cache aprovecha o no la Localidad Espacial y por qué.
b)	<b>Indica</b> para las 3 categorías de fallos de cache (carga, capacidad, conflicto) cuáles aparecen en el código y cuáles no y por qué.
c)	Calcula la cantidad de fallos totales en cache que se producen al ejecutar completamente y una única vez el bucle más interno (bucle marcado en negrita en el código IJK). Debes dar el resultado en función de la constante M.

d)	Calcula la tasa de fallos en cache que se obtiene al ejecutar completamente el bucle más interno 1 vez y da el
_	resultado en porcentaje.
Ļ	
e)	Calcula el total de páginas de Memoria Virtual que se utilizan para ejecutar completamente el bucle más interno 1
	vez. Da el resultado para un valor de M = 512.
Cor	n el objetivo de reducir los fallos de Carga, a la cache de datos anterior se le añade un mecanismo de prefetch para
trae	er a la cache información antes de que sea solicitada en un futuro cercano. Considerar dos tipos de prefetch: (a) se
lana	za un prefetch del bloque i+1 cada vez que se accede al bloque i (prefetch OBL - one block lookahead), y (b) al observar
una	secuencia de accesos a bloque b, b+1*n, entonces se hace prefetch de b+2*n (prefetch con stride).
f)	Calcula la cantidad de fallos totales en cache que se producen al ejecutar completamente el bucle más interno 1
'')	vez en presencia del prefetch OBL. Debes dar el resultado en función de la constante M.
	vez en presencia del prefettiri OBE. Debes dai erresultado en función de la constante ivi.
g)	Calcula la cantidad de fallos totales en cache que se producen al ejecutar completamente el bucle más interno 1
67	vez en presencia del prefetch con stride. Debes dar el resultado en función de la constante M.
	<u> </u>
Sun	poner que se ejecutan completos los tres bucles del código IJK en un procesador con una cache de tamaño limitado y
	a un valor de M = 256. Al pasarle al ejecutable la herramienta valgrind observamos que se ejecutan 691x10 <sup>6</sup>
-	trucciones dinámicas y al instrumentar el código con GetTime ( ) obtenemos un tiempo de ejecución de 57.85 ms.
	emás, el código IJK se modifica implementando una ordenación distinta de sus bucles y añadiendo una optimización
	cional como la que vimos en la práctica 8. El nuevo código se compila con la opción -O3 y se obtiene un ejecutable
	e se ejecuta 12 veces más rápido.
h)	Calcula el rendimiento en MFLOPS del código original y del código optimizado. Da el resultado en GFLOPS.
1	

COGNOMS:																									
NOM:																									
														ļ											
Problema	<b>2.</b> (3,	,4 pı	untos	)																					
Queremos (	estud	iar e	el efec	to de	e la	búsqu	eda y	/ des	scod	ifica	ción	de	inst	rucc	ione	es e	n e	rer	dim	ient	o de	e un	pro	cesac	dor
segmentado programa d puedan cau supondrem	e pru ısar 1	eba os a	P en u cceso	n sim s a c	nulac datos	dor pa s y no	rame os cei	triza ntrar	ble. emo	Para os ex	sim kclus	ıplif siva	icar men	el pr te e	oble n lo	ema os a	, igr	nora sos	rem a in:	os la stru	as int	terfe	eren	cias o	lue
En los proce y el procesa La alternativ taken no inc este proces	dor d va ma curre	lecid ás sir n en	le si el mple d ningu	salto consi: una p	ste e ste e	produ n ejed lizació	cirá (í cutar	takei por i	n) o defe	por e	el co instr	ntra	ario ione	se se s de	egui for	rá e ma	jecı secı	ıtan ıend	do e cial, o	n se con	cue lo q	ncia ue lo	(not os sa	take Itos i	n). not
Con el simu son <i>taken</i> . T ciclos/instru	Гamb	ién l																		•					
a) <b>Calcul</b> a	a el C	PI de	el prod	cesac	lor P	Seq.																			
Ante la perc con el que l mal predich b) <b>Calcul</b> i	nemo o es	s ob de 2	tenido 0 ciclo	o un s. Lo	CPI ( s sal	de 0,9 tos bi	ciclo en pro	s/ins edich	struc nos r	cción	ı. En	el d	diseí	ňo p	ropu	-					-				-
Finalmente, tiene una to considerado que accede instruccione a la cache do de instruccione con Calcula	asa d o los f n a la es eje e inst ones	le ad fallos a cad cuta rucc tien	ciertos s en la che de das. S iones, e una	del cach insta aben es de tasa	95% ne de trucc nos c ecir, de fa	con instr ciones que en se rea allos d	el qu uccio , por P se lizan e 0,0	nes. lo q ejec sólo 5 fal	Los Los que e utan 0,4 a los/a	tiene proc el nú 10x acces acces	e un cesac ímec 10 <sup>9</sup> sos a so y	CP dore ro c inst a la c la p	I de es su de ad rucc cach cach	0,7 ipero ione e po lizac	cicl esca os a es di r ins ión	os/ lare la nán tru es c	instr es le cacl nica cció le 1	en v en v ne e s y q n. Sa 00 c	ión. varia s m ue s aben iclos	Has s in: eno e re nos /fal	sta a struc r qu aliza ader lo.	horaccion e el n 4x nás	a no nes c nun (10 <sup>9</sup> que l	hemada v nero acces	nos vez de sos
-,		2.010	- PC10			u			J., 10			11				, -	. 01			· 141	- 505		-		

El conjunto procesador-cache está conectado a un sistema de memoria principal mediante un único canal de 64 bits al que se ha conectado un DIMM de 4 Gbytes. Este DIMM tiene 8 chips de memoria DDR-SDRAM (Double Data Rate Synchronous DRAM) de un byte de ancho cada uno. El DIMM esta configurado para leer/escribir ráfagas de 64 bytes. La

<sup>1.</sup>La CPU funciona a una frecuencia interna mayor que la memoria por lo que un ciclo de memoria corresponde a múltiples ciclos de CPU.

latencia de fila es de 5 ciclos, la latencia de columna es de 4 ciclos y la latencia de precarga es de 2 ciclos.  d) Suponiendo que el comando ACTIVE se lanza en el ciclo 1, <b>indica</b> en qué ciclo se lanzará el comando PRECHAR en una operación de lectura de un bloque de 64 bytes de la memoria DDR-SDRAM. Justifica la respuesta.
Esta DDR funciona a una frecuencia de 1 GHz. La potencia consumida por la DDR depende de su actividad. Durante acceso a un bloque de 64 bytes se consumen 10 W (desde que se envía el comando ACTIVE hasta que se completa PRECHARGE), mientras que el resto del tiempo consume sólo 1 W. Sabemos que el programa P ha tardado 9 segunden ejecutarse y que se han realizado 0,2x10 <sup>9</sup> accesos de 64 bytes a la DDR.
e) Calcula la potencia media consumida por la memoria DDR durante la ejecución del programa P.
Dado que el rendimiento obtenido hasta ahora es bastante pobre, se ha sugerido hacer prefetch de instrucciones. sugiere hacer prefetch en caso de fallo, de forma que en cada fallo se leerán dos bloques de 64 bytes cada uno: el controlador de memoria envía los comandos necesarios a la DE SDRAM de forma que ambos accesos sean servidos lo más rápidamente posible. Supongamos que los dos bloques est en la misma página de DRAM y que la página no está abierta.
f) Suponiendo que se maximiza el ancho de banda útil del bus y que el comando ACTIVE del primer bloque se lar en el ciclo 1, <b>indica</b> en qué ciclo se lanzará el comando PRECHARGE en una operación de lectura de dos bloque consecutivos de 64 bytes de la memoria DDR-SDRAM. Justifica la respuesta.
Nuestro procesador traduce las instrucciones x86 a microoperaciones RISC (uops). Cada acceso a la cache consume nJ (sea acierto o fallo) y cada instrucción x86 gasta 1 nJ en ser descodificada y traducida a uops.
g) Calcula la energía consumida por la búsqueda y traducción de las instrucciones x86 durante la ejecución de P.
Se ha decidido introducir una cache de micro-ops donde se guardan las uops que ya han sido traducidas para que caso de acierto no sea necesario descodificar y traducir de nuevo las instrucciones x86. Con el simulador se ha medi que para ejecutar P se generan $15x10^9$ uops dinámicas y se realizan $5x10^9$ accesos a la cache de uops. Gracias a esto número de accesos a la cache de instrucciones se ha reducido a $0.4x10^9$ accesos y el número de instrucciones x descodificadas se ha reducido a $1x10^9$ instrucciones. Un acceso a la cache de uops consume $1.6$ nJ.
h) Calcula la energía consumida por el sistema con cache de uops durante la ejecución de P.

COGNOMS:														
NOM:														

## Problema 3. (3,4 puntos)

Una empresa nos ha contratado para analizar su aplicación y proponer mejoras que reduzcan el tiempo de ejecución. Hemos hecho un estudio sobre la aplicación ejecutándola con un solo procesador y un solo disco (un PC de sobremesa con un procesador single-core), y este nos muestra que la aplicación ejecuta de forma iterativa tres fases bien diferenciadas, tal como se ve en la figura:

```
for (;;) {
  LeerDatos();
  ProcesarDatos();
  EscribirResultados();
}
```

- Fase 1: LeerDatos(), únicamente se hacen lecturas desde disco. Ocupa el 10% de toda la ejecución.
- Fase 2: ProcesarDatos(), parte intensiva en cálculo que es totalmente paralelizable. Ocupa el 75% de la ejecución.
- Fase 3: EscribirResultados(), los datos recién calculados se escriben a disco. Ocupa el 15% de la ejecución.

El tiempo de ejecución de la aplicación es de 100 horas. Queremos estudiar diferentes alternativas que nos permitan reducir este tiempo.

a)	<b>Calcula</b> el Speedup máximo que podríamos obtener si ejecutáramos este programa en un supercomputador con un número infinito de procesadores y un disco.
<b>L</b> \	
b)	Calcula con cuántos procesadores tenemos que ejecutar el programa para obtener un Speedup de 3.

Dado que gran parte de la aplicación puede paralelizarse completamente, ofrecemos a la empresa reemplazar el PC anterior por uno nuevo con un procesador multi-core de 12 núcleos.

Además, hemos detectado que el 80% de la Fase 2 se dedica a hacer operaciones matemáticas entre varias matrices en las que cada cálculo es independiente de los demás, así que decidimos transformar el código en esas regiones para utilizar instrucciones SIMD. La arquitectura de los núcleos del procesador nos permite utilizar registros xmm de 128 bits, y el código trabaja con matrices de enteros de 4 bytes.

c)	<b>Calcula</b> el Speedup de la Fase 2 después de aplicar la transformación en el código y de ejecutarlo en el multi-core de 12 nucleos. <b>Calcula</b> el tiempo total de la aplicación después de estas mejoras.

Para mejorar las fases de lectura/escritura de datos necesitaremos un sistema de almacenamiento que nos dé mayor ancho de banda que el actual. Además queremos añadir algún mecanismo de tolerancia a fallos en disco. Valoraremos los sistemas de almacenamiento RAID 10 (con mirror doble), RAID 5 y RAID 6. La aplicación necesita un total de 18 terabytes de almacenamiento.

Para implementar los diversos sistemas RAID disponemos de un único modelo de disco duro de 3TB. d) Calcula cuántos discos duros de 3TB necesitaríamos para cada uno de los niveles de RAID 10 (mirror doble), 5 y 6 si queremos almacenar al menos 18TB útiles Después de explicar las diferencias entre los tres niveles de RAID, la empresa se decanta por montar un RAID5 con 10 discos duros de 3TB. Cada disco duro tiene un MTTF de 100.000 horas, y se calcula que, en caso de fallo, el tiempo necesario para reemplazar un disco físico y reconstruir los datos es de 4 horas. e) Calcula el MTTF del sistema de almacenamiento para nuestro RAID 5 con 10 discos físicos. Por último, la empresa nos pide que calculemos el Speedup de las dos fases de acceso a disco con RAID 5 respecto al original, teniendo en cuenta que en la Fase 1 se leen 7.2 TB y en la Fase 3 se escriben 10.8 TB. Se considera que todos los accesos a disco son secuenciales. f) Calcula el ancho de banda de lectura en la Fase 1 en nuestro sistema RAID 5 con 10 discos físicos. Calcula el ancho de banda de escritura en la Fase 3 en nuestro sistema RAID 5 con 10 discos físicos. h) Calcula el Speedup y el tiempo total de la aplicación respecto al original (1 core + 1 disco) después de aplicar todas las optimizaciones (RAID 5 con 10 discos + 12 cores + SIMD).