Problema 1. Segmentación, riesgos de control.

Uno de los problemas de los procesadores segmentados (y también superescalares y VLIW) es la penalización introducida por los saltos. En los procesadores actuales con muchas etapas pueden transcurrir decenas de ciclos hasta que se calcula la dirección destino del salto y se conoce si el salto será tomado (*taken*) o no (*not taken*).

Como ejemplo supongamos un procesador superescalar segmentado ideal (sin penalizaciones por saltos ni fallos de cache, ni bloqueos debidos a riesgos de datos o estructurales) que es capaz de ejecutar 4 instrucciones por ciclo.

a) **Calcular** el CPI del procesador ideal.

Supongamos ahora que los saltos introducen una penalización de 20 ciclos (manteniendo ideal el resto del procesador).

b) ¿Cuantas instrucciones se dejan de ejecutar cada vez que se ejecuta una instrucción de salto?

La cantidad de saltos que ejecutan los programas no es nada despreciable, por ejemplo, el conjunto de benchmarks SpecInt ejecutan en una arquitectura x86 un 20% de instrucciones de salto. Supongamos que en nuestro procesador ejecutamos un programa con un 20% de saltos.

c) **Calcular** el CPI teniendo encuentra las penalizaciones por salto.
d) **Calcular** cuantas veces más lento se ejecuta el programa por culpa de los saltos respecto el procesador ideal.

Para solventar el problema de los saltos, los procesadores actuales incorporan predictores de saltos en la fase de búsqueda de instrucciones que les permite predecir cual será la siguiente instrucción a ejecutar después de un salto. El procesador ejecuta instrucciones de forma *especulativa* de acuerdo con el predictor de saltos. Cuando finalmente se conoce realmente si el salto salta o no (es decir, se comprueba si el predictor ha acertado o no), es caso de acierto no se ha perdido ningún ciclo (ya se han ejecutado las instrucciones que tocaba), mientras que en caso de fallo se han perdido los ciclos de penalización por salto (ejecutando instrucciones que NO tocaba) y se deben descartar los resultados de las instrucciones ejecutadas de forma *especulativa*.

Supongamos que a nuestro procesador le incorporamos un predictor de saltos que en nuestro programa tienen una tasa de aciertos del 95%.

e) **Calcular** el CPI con el procesador con predictor de saltos.
f) **Calcular** el speedup respecto el procesador sin predictor de saltos.

Problema 2. VLIW, Jerarquia de memória.

Tenemos un procesador VLIW segmentado en el que cada instrucción VLIW puede codificar un máximo de 6 operaciones independientes (cada operación realiza cálculos equivalentes a una instrucción RISC). Dado que en un VLIW una sola instrucción equivale a muchas operaciones, se define una métrica adicional denominada OPC (operaciones por ciclo). Este VLIW sigue el clásico pipeline de 5 etapas:

- F: Búsqueda de la instrucción (*fetch*)
- D: Descodificación y lectura de registros
- E: Ejecución y calculo de direcciones
- M: Acceso a memoria
- W: Escritura de registros (*write*)

Un programa compilado para este procesador VLIW se ha interpretado en un simulador con memoria ideal (entendiendo por ideal el que los accesos a instrucciones se pueden completar durante la etapa F y los accesos a datos se pueden completar durante la etapa M sin necesitar ciclos adicionales). Se han ejecutado 10^9 instrucciones VLIW en 10^9 ciclos con una media de 4 operaciones por instrucción.

a) **Calcular** el IPC y el OPC del programa
b) ¿Cual debería ser el IPC de un procesador RISC superescalar que funciona a la misma frecuencia para obtener el mismo rendimiento.

Analizando los accesos a datos, hemos comprobado que el 20% de las instrucciones VLIW no realizan ningún acceso a memoria, el 40% realiza 1 acceso a memoria y el 40% realiza 2 accesos a memoria simultáneos. Nótese que para obtener el rendimiento anterior es necesario que la cache de datos permita dos accesos por ciclo (además de caches de datos e instrucciones separadas para que no haya riesgos estructurales). Una memoria cache con dos puertos tiene mayor coste (área de chip), mayor tiempo de acceso (que podría repercutir en el tiempo de ciclo) y mayor consumo que una equivalente con un solo puerto. Por ello se desea evaluar el impacto en el rendimiento de usar memorias con un solo puerto (menor coste, tiempo de acceso y consumo). De momento supondremos que no hay fallos de cache.

Memoria cache con un solo puerto y un solo banco. En caso que una misma instrucción VLIW realize 2 accesos a datos simultaneos (esto seria un buen ejemplo de riesgo estructural), el procesador se bloquea durante un ciclo y realiza los accesos en secuencia.

c) **Calcular** los ciclos que tarda el VLIW con un solo banco y un solo puerto
d) **Calcular** el IPC y el OPC

Una alternativa mejor es partir la cache en bancos de un solo puerto de forma que cuando una instrucción VLIW realice 2 accesos a datos simultaneos, solo será necesario bloquear el procesador si ambos accesos acceden al mismo banco. Suponemos que cualquier acceso puede acceder a cualquier banco con la misma probabilidad.

e) **Calcular** la probabilidad de que 2 accesos simultáneos vayan al mismo banco en una cache con 4 bancos.
f) **Calcular** los ciclos que tarda el VLIW con una cache con 4 bancos.
g) **Calcular** el IPC y el OPC.

Problema 3. (3 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 200 horas.

Dado que el coste en tiempo es muy alto, deseamos ejecutar el programa (que en parte es paralelizable) en un sistema multiprocesador con procesadores 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 no puede paralelizarse, ocupa el 10% del tiempo en la ejecución en el PC.

a) **Calcular** 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 1 disco.

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. Además se calcula que el tiempo de sincronización aumenta conforme aumenta el número de procesos por lo que hay un límite a la cantidad de procesadores que podemos usar ganando velocidad. Suponiendo que el tiempo de cálculo de la Fase 2 se puede dividir de forma perfecta entre los procesadores que se usan pero que el tiempo de sincronización aumenta en un 0,5% del tiempo de ejecución en el PC por cada procesador.

b) **Calcular** la expresión matemática del tiempo de ejecución total del programa en función del número de procesadores (N).

A partir de la ecuación resultado del apartado anterior nos interesa encontrar la cantidad de procesadores (N) que resulta en un tiempo de ejecución mínimo. Para ello podemos derivar la ecuación e igualar a 0 para obtener sus mínimos y máximos. A continuación basta con probar para estos valores de N cual es el que da el tiempo total mínimo en la ecuación original.

c) Deriva la ecuación anterior, igualala a cero y calcula el número de procesadores (N) en los que el tiempo de ejecución total del programa es mínimo.

Del resultado anterior sabemos que el tiempo de ejecución de la Fase 2 en cada procesador (sincronización incluida) es de aproximadamente 26 h.

d) **Calcular** cual es la ganancia en tiempo real del programa cuando se ejecuta en un entorno con el número ideal de procesadores y un disco.

Nuestro supercomputador, además de poseer muchos procesadores también dispone de un RAID de discos. Esto nos permite suponer que la Fase 3 se realizará más rápidamente 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.

e) **Calcular** la ganancia en tiempo con respecto a un PC de sobremesa que obtendríamos en el programa si lo ejecutaramos en un entorno con un solo procesador y un RAID 0 formado por 10 discos.

Nuestro sistema supercomputador está compuesto por un RAID 5 de 10 discos, cada uno de los cuales es igual al del PC de sobremesa. Sabemos que en nuestro programa el tiempo de la Fase 3 (es decir, el tiempo donde se trabaja con los discos) se gasta la mitad en escrituras y la mitad en lecturas. A partir de aquí suponemos que siempre que hablamos del supercomputador nos referimos a la configuración con el número ideal de procesadores y un RAID 5 de 10 discos.

f) **Calcular** cuanto tiempo tardará la Fase 3 de nuestro programa ejecutada en el supercomputador.

g) **Calcular** la ganancia en tiempo del supercomputador con respecto al PC de sobremesa.

Sabemos que el programa contiene $648 * 10^{13}$ instrucciones dinámicas de las cuales $216 * 10^{13}$ son instrucciones de coma flotante que realizan un total de $72 * 10^{13}$ operaciones de coma flotante.

h) **Calcular** a cuantos MIPS y MFLOPs se ejecuta el programa en el PC de sobremesa .

Al pasar el programa al supercomputador medimos que las instrucciones dinámicas de sincronización son $13 * 10^{13}$ en total.

i) **Calcular** a cuantos MIPS y MFLOPs se ejecuta el programa en el supercomputador.

Sabemos que la potencia consumida por cada procesador (tanto del PC de sobremesa como del supercomputador, que son iguales) es de 90 W, mientras que la potencia consumida por cada disco (también son todos iguales) es de 30 W. Suponemos que estos son los únicos elementos con consumo significativo y que se encuentran funcionando durante toda la ejecución del programa

j) **Calcular** los MFLOPS/W tanto del PC de sobremesa como del supercomputador.

k) **Calcular** el incremento de eficiencia energética que podríamos obtener para el supercomputador si consiguiéramos apagar completamente los elementos que no se utilizan (es decir, 12 de los 13 procesadores durante las etapas de cálculo secuencial y los 10 discos durante todas las etapas sin E/S).