ESTRUCTURA DE COMPUTADORES

27 de enero de 2021

APELLIDOS: _____ **NOMBRE:**_____

1. **[1p]** Un procesador implementa un camino de datos segmentado con 10 etapas. Cada una de las etapas se completa en un ciclo de reloj. En este procesador todas las instrucciones pasan por todas las etapas, y las instrucciones de salto condicional (bne, beq) son las únicas que pueden provocar un riesgo. Cuando una instrucción de salto condicional termina de ejecutar la etapa 1, el *pipeline* se bloquea hasta que se evalúa la condición de salto.

   Sobre este procesador ejecutamos un código que tarda 1809 ciclos tras la ejecución efectiva de 1000 instrucciones, de las cuales 200 son instrucciones de salto condicional y 400 son operaciones en punto flotante.

   a) **[0.2p]** Determina en qué etapa del camino de datos se evalúa la condición de salto.

   b) **[0.4p]** Calcula cuál de las siguientes acciones aplicadas individualmente mejora el rendimiento del procesador en mayor medida:

   - Reducir la longitud del *pipeline* en 2 etapas, sin que afecte a los riesgos que se pudiesen producir.
   - Mover la evaluación de la condición de salto a la etapa 4
   - Utilizar una estrategia de predicción de salto que en el código descrito tiene una tasa de acierto de un 20 %
   - Utilizar una estrategia de salto retardado, sabiendo que en término medio disponemos de 1,5 instrucciones para rellenar el hueco de retardo (el resto se rellena con instrucciones *nop*).

   c) **[0.4p]** Al compilar el código fuente que produjo el código máquina anterior con nuestro nuevo compilador, al que hemos demoninado *ecc*, obtenemos un código de 1400 instrucciones efectivas de las que 600 son operaciones en punto flotante, que tarda 2200 ciclos en ejecutarse. Buscamos una métrica de rendimiento con la que mostrar que *ecc* supera al compilador anterior.

   - Determina con qué métrica podríamos alardear más de nuestro nuevo compilador: tiempo de ejecución, MIPS o GFLOPS.
   - Desde el punto de vista de un potencial usuario de *ecc*, ¿sería una comparación justa?

2. [**1p**] Los códigos de tipo "streaming", entre los que se encuentran los reproductores de vídeo o audio, se caracterizan por consumir grandes cantidades de datos, pero no reusarlos. Considérese el siguiente código:

```
int A[N];
for( i = 0; i < N; ++i ) {
    r += A[i];
}
```

Contesta a las siguientes preguntas:

a) Asumiendo una cache de correspondencia directa de 64 kB con tamaño de línea de 32 bytes, y que el tamaño de un `int` es de 4 bytes, ¿cuál será la tasa de fallos de este código? ¿Qué tipo de fallos es el más común en este código?

b) ¿Cómo varía la tasa de fallos al variar el tamaño del array? ¿Y al variar el tamaño de la cache?

c) ¿Cómo varía la tasa de fallos para tamaños de línea de 16, 64 y 128 bytes?

d) Supongamos otro código que presenta las siguientes tasas de fallo según el tamaño de bloque:

8: 4 %	16: 3 %	32: 2 %	64: 1.5 %	128: 1 %

¿Cuál es el tamaño de línea L óptimo para una latencia de acceso de 20*L ciclos?

3. [**1p**] Considera un computador con un sistema de memoria virtual paginado, una TLB, una única caché y que soporta varios procesos. La memoria es direccionable a nivel de byte y el tamaño del espacio virtual es de 256 TB. La TLB tiene 128 entradas y es totalmente asociativa. La caché, de 4 MB, es asociativa por conjuntos de 8 vías y el tamaño de línea es de 64 B.

a) [**0,3p**] Calcula el tamaño de página mínimo que permitiría aplicar la técnica VIPT (índice virtual, etiqueta física) en este sistema.

b) [**0,1p**] ¿Cuántas páginas virtuales se soportarían por proceso?

c) [**0,6p**] Justifica razonadamente si las siguientes afirmaciones son verdaderas o falsas:

- Aumentar el tamaño de página haría que se redujese el tamaño de la tabla de páginas.
- Aumentar el tamaño de página haría que se redujese la fragmentación interna.
- Duplicar el tamaño de la TLB impediría aplicar la técnica VIPT.
- Sería posible seguir utilizando la técnica VIPT en un sistema de memoria virtual segmentada en el que la longitud máxima de segmento fuese el tamaño de página calculado anteriormente.
- Sería eficiente aplicar la técnica de escritura directa (write-through) en el sistema de memoria virtual.
- Nunca sería útil tener un espacio de memoria física mayor que el espacio de memoria virtual.

4. [**1p**] Sea un computador con las siguientes características:

   - Las direcciones de memoria y las palabras son de 64 bits.
   - Tiene una jerarquía caché de dos niveles. El primer nivel tiene una tasa de fallo del 10 % y un tiempo de acierto de 5 ns. El segundo nivel tiene un tiempo de acierto de 8 ns y una tasa de fallo del 40 %. El tiempo medio de acceso a todo el sistema de memoria es de 9 ns cuando ambos niveles tienen un tamaño de línea de 8 palabras.
   - La caché de segundo nivel se conecta mediante un bus síncrono de 64 bits a la memoria principal, en el que tanto una transferencia de 64 bits como el envío de la dirección de memoria requieren 1 ciclo de reloj.
   - El sistema de memoria y el bus soportan accesos a bloques de 8 palabras de 64 bits. Se necesitan dos ciclos de reloj entre dos operaciones de bus (se supondrá el bus libre antes de cada acceso). Esto es, entre el envío de dos bloques de palabras, el bus esperará 2 ciclos. La memoria tarda lo equivalente a 20 ciclos del bus en acceder a las primeras 4 palabras del bloque; y luego cada grupo adicional de cuatro palabras del mismo bloque se lee en 13 ciclos de bus. Se considerará que la transferencia de los datos por el bus puede solaparse con la lectura de las cuatro palabras siguientes.

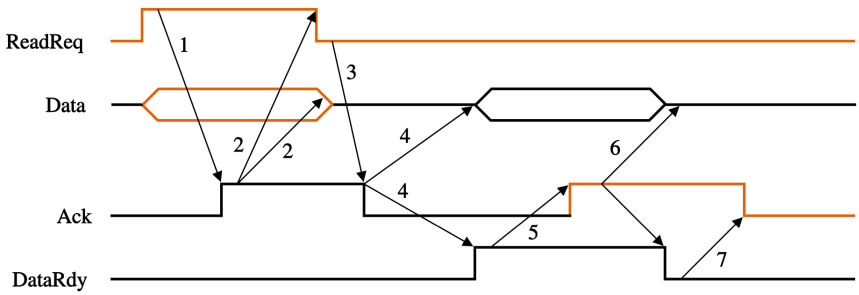
   Calcular:

   a) [**0.5p**] La frecuencia del bus síncrono que comunica la caché de segundo nivel con la memoria principal. El ancho de banda y la latencia para la lectura, desde la caché de segundo nivel, de 128 palabras de memoria, en el caso de transferencias de bloques de 8 palabras.

   b) [**0.3p**] Supongamos que este bus soporta transacciones de bloques de 4 palabras y que la penalización por fallo de la caché de segundo nivel es de 54 nanosegundos cuando trabaja con líneas de 4 palabras. ¿Tendría sentido reemplazar este bus síncrono de memoria por un bus asíncrono donde cada paso del protocolo *handshaking* requiere de 30 nanosegundos en transferencias de bloques de 4 palabras? ¿Por qué? Justifica numéricamente tu respuesta. ¿En el caso general tiene sentido reemplazar un bus de memoria síncrono por uno asíncrono?

   c) [**0.2p**] Este computador tiene un total de 1024 GB de información muy valiosa almacenada en disco. Dispongo de hasta 1000 euros para montar un RAID y sabiendo que cada disco de 256 GB me cuesta 100 euros, ¿qué configuración RAID escogería?

   Puedes ayudarte de la siguiente figura para el apartado de bus asíncrono:



SOLUCIÓN

1. *a*) Si no hubiese ningún riesgo, el código se ejecutaría en 1009 ciclos (10 ciclos de la primera instrucción y un ciclo adicional por cada una de las 999 instrucciones restantes). Por lo tanto los $1809 - 1009 = 800$ ciclos restantes están provocados por las 200 instrucciones de salto, a razón de 4 ciclos cada una, lo que quiere decir que la condición de salto se evalúa en la etapa $1 + 4 = 5$. Lo podemos ver facilmente con un diagrama:

salto	E1	E2	E3	E4	**E5**	E6	E7	...
siguiente		-	-	-	-	E1	E2	...

Otras respuestas correctamente razonadas también fueron consideradas como válidas.

b)
- Reducir el *pipeline* en 2 etapas sólo nos aporta una mejora de 2 ciclos, por lo que descartamos esta opción.
- Si el salto se decide en la etapa 4, cada instrucción de salto provocará un bloqueo de 3 ciclos en lugar de 4. La aceleración es $\frac{1809}{1009+200*3} = \frac{1809}{1609} = 1{,}12$
- Si aplicamos el predictor de salto y acierta el 20 %, significa que de las 200 instrucciones de salto habrá 40 aciertos en los que no se producirá un riesgo, y 160 en las que el *pipeline* se detendrá durante 4 ciclos. El número de ciclos tras aplicar esta mejora es $1009+160*4 = 1649$, y la aceleración es $1809/1649 = 1{,}097$
- Utilizando salto retardado se reduce el riesgo provocado por cada salto en 1,5 ciclos, por lo que ahora el riesgo en cada salto será de $4 - 1{,}5 = 2{,}5$ ciclos. La aceleración es $\frac{1809}{1009+200*2{,}5} = \frac{1809}{1509} = 1{,}20$. Ésta es la opción más beneficiosa.

c)
- El tiempo de ejecución (que sería lo más justo) no nos sirve porque con el nuevo compilador el código tarda más ciclos en ejecutarse. La máquina en la que ejecutamos los códigos es la misma, porque estamos comparando los compiladores, y por lo tanto el tiempo de ciclo (T_c) es constante y el tiempo de ejecución es proporcional al número de ciclos.
- Con la métrica MIPS, aunque no podemos calcular el valor de MIPS exacto de cada código porque desconocemos el tiempo de ciclo, comparando ambos códigos podemos calcular que éste es $\frac{1400/(2200\times\cancel{T_c}\times\cancel{10^6})}{1000/(1809\times\cancel{T_c}\times\cancel{10^6})} = \frac{1400/2200}{1000/1809} = 1{,}15$ veces mayor.
- Lo mismo sucede con la métrica GFLOPS. No podemos calcular su valor exacto, pero el valor del nuevo compilador es $\frac{600/(2200\times\cancel{T_c}\times\cancel{10^9})}{400/(1809\times\cancel{T_c}\times\cancel{10^9})} = \frac{600/2200}{400/1809} = 1{,}23$ veces mayor.
- Por lo tanto la mejor opción es usar la métrica GFLOPS, aunque no sería una comparación justa ya que el compilador original genera un código más eficiente y lo que buscamos en un compilador es que produzca un código que se ejecute lo más rápido posible.

2. *a*) Dado que el tamaño de línea es de 32 bytes y cada entero ocupa 4 bytes, cada lectura de una línea de memoria traerá 8 enteros a cache. Asumiendo que `r` e `i` están almacenadas en registros del micro y que la dirección de inicio de `A` está alineada a una palabra de memoria, y teniendo en cuenta que no hay reuso y que el recorrido del array es secuencial, tendremos un fallo forzoso por cada 8 accesos, resultando una tasa de fallos de $\frac{1}{8}$. Todos los fallos serían forzosos, no es posible que existan fallos de capacidad ni de conflicto al no haber reuso.

b) Ni el tamaño del array ni el de la cache influyen. Lo único que afecta a la tasa de fallos de este código es el tamaño de la línea cache y el tamaño de cada elemento del array.

c) Para 16, 64 y 128 bytes podríamos almacenar, respectivamente, 4, 16, y 32 enteros en cada línea cache. Las tasas de fallos serían, por tanto, de $\frac{1}{4}$, $\frac{1}{16}$, y $\frac{1}{32}$, respectivamente.

d) Asumiendo que el tamaño de línea no influye en el tiempo de acierto, la parte del tiempo medio de acceso a memoria correspondiente a la penalización por fallo en cada uno de los casos será:

- Líneas de 8B: $8 \cdot 20 \cdot 0{,}04 = 6{,}4$ *ciclos*
- Líneas de 16B: $16 \cdot 20 \cdot 0{,}03 = 9{,}6$ *ciclos*
- Líneas de 32B: $32 \cdot 20 \cdot 0{,}02 = 12{,}8$ *ciclos*
- Líneas de 64B: $64 \cdot 20 \cdot 0{,}015 = 19{,}2$ *ciclos*
- Líneas de 128B: $128 \cdot 20 \cdot 0{,}01 = 25{,}6$ *ciclos*

Por tanto, en este caso el mejor tamaño de línea es el de 8B, dado que su incremento no mejora la tasa de fallos lo suficiente como para compensar el incremento en la penalización de fallo.

3. *a*) $|V| = 256\ TB = 2^{48}B \Rightarrow$ DV de 48 bits
$|C| = |ZA| = 4\ MB = 2^{22}B$
$|l| = 64\ B = 2^6 B \Rightarrow \Delta_l$ de 6 bits
#líneas $= |ZA|/|l| = 2^{22-6} = 2^{16}$
#conjuntos = #líneas / 8 vías $= 2^{16-3} = 2^{13} \Rightarrow$ índice de 13 bits

Núm. PV	Δ_p
48-p bits	p bits

Núm. PF	Δ_p
??-p bits	p bits

Etiqueta	Índice	Δ_l
?? bits	13 bits	6 bits

Para poder aplicar la técnica VIPT, el índice que utilizamos para acceder a la caché tiene que coincidir dentro de los bits del desplazamiento de página (la parte de la DV que no se traduce). Por tanto, $p \geq (13 + 6) \Rightarrow |P| \geq 2^{19}B$.

b) Cada proceso tiene su propio espacio de direcciones virtuales. Nos quedan hasta $48 - 19 = 29$ bits para expresar el número de página virtual. Por tanto, #PV $\leq 2^{29}$.

c)
- VERDADERO. Al aumentar el tamaño de página estamos reduciendo el número de páginas del sistema, por lo que la tabla tendrá menos entradas (por tener menos páginas virtuales) y estas entradas serán de menor tamaño (ya que necesitaremos menos bits para guardar el número de página física al que traducimos puesto que también tendremos menos páginas físicas).
- FALSO. Al aumentar el tamaño de página estamos aumentando la probabilidad de tener espacio desaprovechado en cada una de ellas.
- FALSO. La TLB es una caché totalmente asociativa que guarda las últimas traducciones. Así, contiene pares {página virtual, página física} (junto con información adicional de control). Por tanto, duplicar su tamaño simplemente nos permitirá tener más pares. No tiene influencia en la aplicación de la técnica VIPT, como pudimos ver en el apartado (a).
- FALSO. Si bien es cierto que el desplazamiento dentro del segmento no se traduce, en el proceso traducción éste se suma a la dirección física donde comience

el segmento en cuestión. Por tanto, dichos bits de la dirección virtual no podemos utilizarlos para indexar la caché ya que, típicamente, no coincidirán con los últimos bits de la dirección física a la que se traduce.

- FALSO. En escritura directa llevaríamos cada escritura en memoria principal al nivel inferior. El nivel inferior en la jerarquía es el disco duro. Sea de estado sólido o magnético con acceso rotacional, su tiempo de acceso es varios órdenes de magnitud mayor.
- FALSO. En un sistema que soporta varios procesos, como el que estamos contemplando, nos permitiría tener en memoria física más páginas de los distintos procesos que hiciesen los cambios de contexto más rápidos. Otro ejemplo sería el uso de varias máquinas virtuales en un equipo.

4. *a*) Si tiempo medio acceso son 9 ns $= 5 + 0{,}1 \times (8 + 0{,}4 \times PF)$, por tanto $PF = 80ns$
Tenemos 1 ciclo enviar $dir + 20$ ciclos para acceder pal 0-3 + max(4 ciclos de envío pal 0-3, 13 ciclos acceso pal 4-7) + 4 ciclos envío pal 4-7 + 2 ciclos espera = 40 ciclos
Si 40 ciclos tardan 80 ns (PF) entonces $T_{ciclo} = 2$ ns y por tanto la frecuencia es su inversa: $f = 500MHz$

   Si 1 transacción son 40 ciclos y hay 16 transacciones (128 palabras / 8 lineas) la latencia es de 16×40 ciclos = 640 ciclos, que son 1280 ns. El ancho de banda se calcula como $\frac{128 \times 8 \ B}{1280 \times 10^{-9} \ s} = 800$ MB/s

   b) Podemos calcular el tiempo de ciclo de nuevo si hacemos que $54 \ ns = 27 \ ciclos$ y nos vuelve a salir $T_{ciclo} = 2 \ ns$; o podemos conservar el cálculo del anterior apartado. En el caso de este asíncrono salen $30 + max(3 \times 30, 54) + 3 \times 30 = 210$ ns entonces no es más rápido. Se espera una justificación sobre el uso de buses síncronos para el intercambio de datos entre dispositivos rápidos.

   c) Como el enunciado me dice que los datos son muy valiosos, la configuración más adecuada es un RAID 1 (preferiblemente 1+0) aunque sacrifique rendimiento.