ESTRUCTURA DE COMPUTADORES 2 de julio de 2021

APELLIDOS: _____ **NOMBRE:** _____

[1p] 1. A continuación se muestra un **fragmento** del diagrama multiciclo que resulta de la ejecución de un código en un procesador MIPS de 5 etapas como el estudiado en clase. En este diagrama no se señalan explícitamente los riesgos ni las anticipaciones. En el código existen instrucciones anteriores y posteriores a este fragmento que desconocemos.

		1	2	3	4	5	6	7	8	9	10	11	12
1	loop: `lw $t0, 0($a0)`	IF	ID	EX	MEM	WB							
2	`lw $t1, 4($a0)`		IF	ID	EX	MEM	WB						
3	`add $t2, $t2, $t0`			IF	ID	–	EX	MEM	WB				
4	`addi $a0, $a0, 8`				IF	–	ID	EX	MEM	WB			
5	`sw $t1, 0($a1)`						IF	ID	EX	MEM	WB		
6	`sw $t2, 4($a1)`							IF	ID	EX	MEM	WB	
7	`bne $a1, $a0, loop`								IF	ID	EX	MEM	WB

(a) **[0,4p]** Razona (cuando sea posible) si este procesador incorpora las siguientes técnicas estudiadas en clase. En caso afirmativo, indica cuáles de ellas incorpora.

- Unidad de detección de riesgos
- Unidad de anticipación a la etapa EX
- Resolución de salto en la etapa ID
- Salto fijo no efectivo

(b) **[0,3p]** Dadas las instrucciones del código anterior, razona si podemos reordenarlas de forma que ocurran riesgos de control, de datos y estructurales, aunque se modifique la lógica original del código.

(c) **[0,3p]** Si el procesador implementa la técnica de salto retardado, determina qué instrucciones podemos mover con seguridad al hueco de retardo, indicando también posibles modificaciones necesarias, y razona cuál sería la mejor opción.

[1p] 2. Un computador dispone de una caché de un único nivel asociativa por conjuntos de 4 vías, de 64 kB en total, con tamaño de línea de 16 bytes.

Considérese el siguiente código:

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

Sabiendo que el tamaño de un `int` es de 4 bytes, que las variables escalares se almacenan en registros del micro, y que tanto N como M son potencias de 2, contesta a las siguientes preguntas:

(a) **[0.25p]** Calcula el valor de M para que cada acceso al array en una nueva iteración del bucle interno vaya a parar al mismo conjunto caché que en la iteración anterior. Para dicho valor de M, calcula el valor de N de modo que no se produzcan fallos de conflicto en el acceso al array completo.

(b) **[0.25p]** Partiendo de los valores calculados de N y M, deduce la relación entre ambos para garantizar un acceso al array libre de fallos de conflicto. Recuerda que tanto N como M se suponen potencias de 2.

(c) **[0.25p]** ¿Qué ocurre si duplicamos la asociatividad, pero manteniendo el tamaño total de la caché constante?

(d) **[0.25p]** Calcula la tasa de fallos para M=2048, N=16.

[1p] 3. Considera un computador con un sistema de memoria virtual segmentado. El tamaño del espacio virtual es de 16 TB y el tamaño del espacio físico es de 4 GB. El tamaño máximo de segmento es de 2 GB. En un momento dado, los siguientes segmentos están residentes en memoria física:

- Segmento 0x11, de 256 MB y que comienza en la dirección 0x00000000.
- Segmento 0x22, de 1 GB y que comienza en la dirección 0x20000000.
- Segmento 0x33, de 512 MB y que comienza en la dirección 0x80000000.
- Segmento 0x44, de 256 MB y que comienza en la dirección 0xD0000000.

(a) **[0,2p]** Dibuja un esquema que represente el estado de la memoria física del sistema, indicando todos los segmentos residentes.

(b) **[0,3p]** Diseña razonadamente una posible estructura de la tabla de segmentos de este sistema. Ejemplifica su funcionamiento traduciendo la dirección virtual 0x1987654321.

(c) **[0,3p]** Se reciben, secuencialmente, peticiones para ubicar los siguientes segmentos:

- Segmento 0xAA, de 256 MB.
- Segmento 0xBB, de 576 MB.

Explica detalladamente cómo se atenderían con el algoritmo Best-Fit.

(d) **[0,2p]** Este sistema de memoria virtual, ¿presenta fragmentación interna? ¿Y fragmentación externa? Justifica tu respuesta apoyándote en los apartados anteriores.

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

- Las direcciones de memoria y las palabras son de 32 bits.
- Tiene un sistema de memoria que permite accesos en bloques de 4, 8 y 16 palabras.
- Tenemos dos buses candidatos para conectarse a este sistema de memoria, ambos son síncronos, de 32 bits y tienen la misma frecuencia de reloj. El bus A está configurado para soportar accesos a memoria en bloques de 8 palabras de 32 bits y no requiere de ningún ciclo de espera entre dos operaciones de bus, es decir, entre el envío de dos bloques de palabras no hay espera alguna. En cambio, el bus B solo soporta accesos a memoria en bloques de 16 palabras de 32 bits y además necesita esperar un número de ciclos entre el envío de dos bloques de palabras (se supondrá el bus libre antes de cada acceso).
- Para cada bloque, 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 10 ciclos de bus. Se considerará que la transferencia de los datos leídos más recientemente por el bus puede solaparse con la lectura de las cuatro palabras siguientes.

Calcular:

(a) **[0.65p]** El número de ciclos de espera que tendrá que realizar el Bus B en cada transacción para que el ancho de banda del Bus A sea igual al ancho de banda del Bus B cuando se leen 128 palabras de memoria.

(b) **[0.35p]** Este computador tiene un total de 1024 GB de información almacenada en 4 discos de 256 GB en RAID 0. Me dispongo a montar un sistema RAID para tener cierta de tolerancia a fallos, pero que me ofrezca concurrencia a la hora de acceder a los datos. ¿Qué configuración RAID escogería y cuántos discos requeriría? Hay una oferta de un proveedor que me ofrece a precio muy económico hasta 8 discos de la misma serie de fabricación. ¿Es buena idea?

1. ————————

(a) Hay unidad de detección de riesgos, porque de lo contrario no podríamos tener la burbuja en la instrucción 3. No hay unidad de anticipación a la etapa EX, porque entonces el riesgo anterior no se habría producido al poder anticipar $t0 a la instrucción 3. La resolución de salto en la etapa ID y salto fijo no efectivo no lo podemos saber porque cuando el salto ejecuta su etapa ID los datos que necesita ya están actualizados en el banco de registros y por tanto no sabemos si la condición de salto se evalúa en ID o en alguna etapa posterior, y además en el diagrama no se muestra qué ocurre después del salto.

(b) El código original ya presenta un riesgo de datos, en concreto un riesgo RAW entre las instrucciones 1 y 3. La instrucción 6 conlleva un riesgo de control tras su ejecución salvo que implemente salto retardado, tanto si se implementa una predicción de salto efectiva como no efectiva. Con las instrucciones disponibles no es posible forzar un riesgo estructural, porque todas tienen la misma latencia y por lo tanto nunca coincidirán en la misma etapa.

(c) Las instrucciones 1, 2 y 4 se podrían mover "desde destino de salto" con cierta modificación, pero eso implicaría que se ejecuten una vez más al finalizar el bucle. Como desconocemos la estructura de los datos reservados en memoria y el código que se ejecuta tras el bucle, no es un movimiento seguro. La instrucción 3 no puede moverse por dependencias con instrucciones anteriores y posteriores. Las instrucciones 5 y 6 pueden moverse con seguridad puesto que no tienen dependencias con las instrucciones posteriores. Si el salto se decide en ID, cosa que no sabemos, se crearía un riesgo entre las instrucciones 4 y 7 (al no poder anticipar el valor de $a0) y no se obtendría ningún beneficio.

2. ————————

(a) $T_{cache} = 2^{16}B, T_{linea} = 2^4B, \#vias = 2^2$: Hay $2^{16}/2^4/2^2 = 2^{10}$ conjuntos. En cada iteración saltamos M elementos de 2^2 Bytes: $2 \times M$ Bytes. Para coincidir de nuevo en el mismo conjunto, el salto debe ser de 2^{10} conjuntos $\times 2^4$ Bytes/linea $= 2^{14}$ Bytes. Como cada elemento de A ocupa 2^2 Bytes $\rightarrow M = 2^{12} = 4096$.

Como hay 4 vías, podemos repetir el mismo conjunto 4 veces sin que ocurra un fallo de conflicto, por lo que $N \leq 4$.

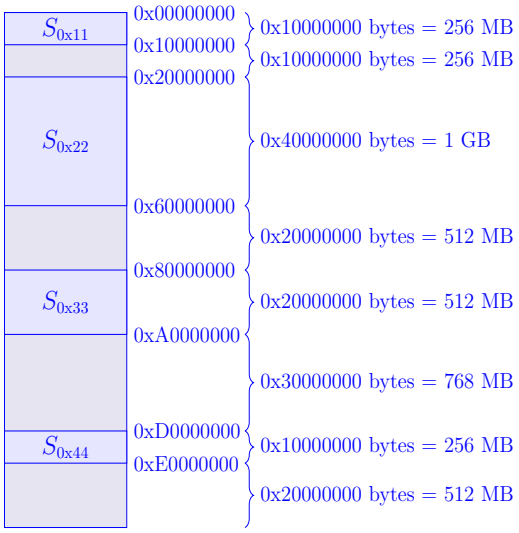
(b) Partiendo de los valores anteriores, si se reduce M a la mitad se alternarán el uso de 2 conjuntos (8 vías). Si se vuelve a reducir a la mitad, 4 conjuntos (16 vías), etc. Por lo tanto al reducir M podemos incrementar N proporcionalmente manteniendo la tasa de fallos. Entonces, la relación entre M y N debe ser $M = 2^k, N \leq max(4, 2^{14-k})$. Es decir, si M pasa de 4096 no afecta a N, que debe mantenerse en un valor menor o igual que la asociatividad.

(c) Se permiten valores mayores de N, en concreto $N \leq max(8, 2^{13-k})$.

(d) Dado que los valores $(M = 2^{11}, N = 2^4)$ no cumplen lo calculado anteriormente, la tasa de fallos será del 100%. Como explicación adicional, con $M = 2^{11}$ se repite el mismo conjunto cada 2 iteraciones. En las 8 primeras iteraciones del bucle interno se llenarán los 2 conjuntos accedidos completamente, y en las 8 iteraciones restantes se sobreescribirán los datos de la caché, y de esa forma no podrán ser reutilizados en el acceso a la segunda columna de la matriz A.

3.

(a)



(b) La estructura de cada entrada podría ser:

Residencia (1)	Otros bits de control (c)	Dirección base (32)	Tamaño (31)

La dirección base es una dirección física completa: $|F| = 4\ GB = 2^{32}\ B$
La entrada "Tamaño" está condicionado por el tamaño máximo de segmento: $max|S| = 2\ GB = 2^{31}\ B$
Deberíamos escoger un número de bits c tal que el tamaño total de la entrada fuese un número entero de bytes (es decir, $c = 0, 8, 16 \ldots$).
$|V| = 16\ TB = 2^{44} B \Rightarrow$ DV de 44 bits

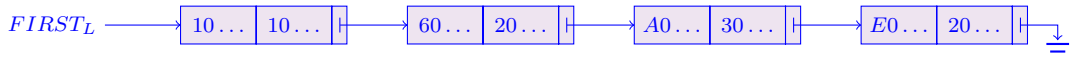
nSV	Δ_s
13 bits	31 bits

Diviendo los bits de 0x1987654321 tenemos que nSV = 0x0033 y Δ_s = 0x07654321. El SV 0x33 es uno de los segmentos que mencionan en el enunciado como residentes. La entrada correspondiente en la tabla sería:

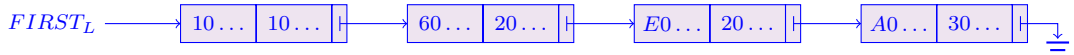
Residencia (1)	Otros bits de control (c)	Dirección base (32)	Tamaño (31)
1	–	0x80000000	0x20000000

Tenemos que comprobar también si el desplazamiento está dentro del tamaño del segmento: Δ_s = 0x07654321 $\leq$ 0x20000000.
Con lo que la direeción física a la que se traduce será DF = 0x80000000 + 0x07654321 = 0x87654321

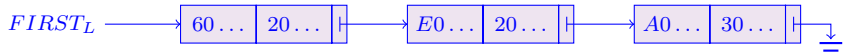
(c) La lista enlazada del sistema sería:



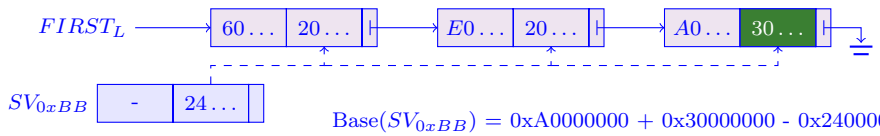
El algoritmo Best-Fit busca el espacio libre más pequeño con el tamaño suficiente. Por ello, trabaja con la lista ordenada por tamaño creciente de espacio libre.



El segmento 0xAA tiene tamaño $|SV_{0xAA}| = 256\ MB = 2^{28}\ B = 0x10000000\ B$, por lo que se asigna al primer espacio libre de la lista. La lista queda como:



El segmento 0xBB tiene tamaño $|SV_{0xBB}| = 576\ MB = 0x24000000\ B$, por lo que se asigna al último espacio libre de la lista. La lista queda como:



Base(SV_{0xBB}) = 0xA0000000 + 0x30000000 - 0x24000000
Base(SV_{0xBB}) = 0xAC000000
Tamaño(Q) = 0x30000000 - 0x24000000
Tamaño(Q) = 0x0C000000

(d) No presenta fragmentación interna: los segmentos virtuales tienen tamaño variable para ajustarse a los datos/código que sea necesario.

En cambio, sí que presenta fragmentación externa: es común que vayan quedando espacios libres en memoria de tamaño tan pequeño que no sean útiles. Un ejemplo de esta situación sería el nuevo nodo Q de 0x0C000000 bytes que se ha generado en el apartado anterior.

4. ———————

(a) BUS A:

Son 128/8 = 16 transacciones donde cada una tiene 1 ciclo para el envío de la dirección + 20 ciclos para las primeras cuatro palabras + max(4 ciclos, 10 ciclos) + 4 ciclos para enviar las últimas 4 palabras = 35 ciclos. En 16 transacciones son 560 ciclos.

BUS B:

Son 128/16 = 8 transacciones donde cada una tiene 1 ciclo env. + 20c primeras cuatro + max(4c,10c) + max(4c,10c)+max(4c,10c)+4c env. últ. 4 pal + Tespera = 55 + Tespera ciclos. En 8 transacciones son 440 + 8 Tespera ciclos.

Para que $AB_A == AB_B$ entonces el número ciclos de Bus A tiene que ser igual que el del Bus B:

$560 = (55 + T_{esp}) \cdot 8$ y por tanto $T_{esp} = 15$ ciclos.

(b) RAID 4, 5 y 6 son la mejor opción donde se necesitan uno, uno y dos discos extras respectivamente. Sin embargo, RAID 4 forma un cuello de botella al no tener la redundancia distribuída por lo que RAID 5 y RAID 6 son mejores opciones. Para presentar mayor tasa de tolerancia a fallos, es mejor quedarnos con RAID 6 al tener hasta 2 discos de recuperación.

No es buena idea porque empíricamente se demuestra que es más probable que de haber un fallo en un disco, todos los de su serie de fabricación lo contengan igualmente, lo que nos reduce la vida esperada del sistema RAID.