ESTRUCTURA DE COMPUTADORES

Segundo examen parcial **23 de marzo de 2015**

Apellidos y nombre	DNI	Grupo

1 ***(1 punto)*** Una CPU tiene una capacidad de direccionamiento de 16 GB y emplea palabras de 64 bits.

a) ¿Cuántos bits de dirección son necesarios para direccionar toda la memoria? Razona la respuesta

34 bits, ya que 2^{34} = 16 G

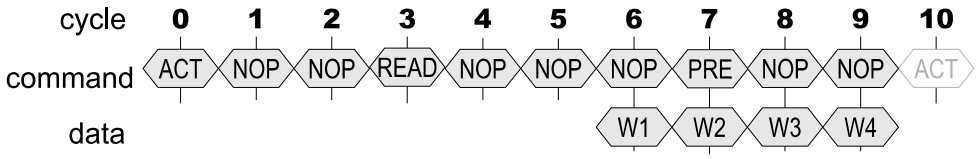
b) ¿Cuántas líneas de selección de octeto tiene la CPU (BE_i*)? Razona la respuesta

8 líneas, desde BE_0* hasta BE_7*, ya que 64 bits = 8 bytes

c) ¿Cómo se denominan las líneas físicas de dirección de la CPU (A_i)? Razona la respuesta

Desde A_3 hasta A_{33}, ya que los bits A_0..A_2 están descodificados por BE_0*..BE_7*

2 ***(1 punto)*** El cronograma de la figura corresponde a una operación de lectura de un bloque de 4 palabras de un chip de memoria SDRAM que funciona a 100 MHz y tiene un ancho de palabra de 16 bits.



Indica:

a) Cuál es el **tiempo de acceso** de esta memoria, expresado en nanosegundos.

Como los datos tardan en aparecer 6 ciclos de 1 / 100MHz = 10 ns, el tiempo de acceso es de 60 ns

b) Cuál es su **ancho de banda**.

1 palabra de 2 B (16 bits) cada ciclo de reloj (10 ns) ➔ 2 B / 10 ns = 200 MB/s

3 ***(1 punto)*** Indica en cuáles de los siguientes casos sería posible concatenar el acceso a dos bloques de un chip de memoria RAM dinámica:

a) Dos bloques de una misma fila en un mismo banco.	☐SÍ	☐NO	**(SÍ)**
b) Dos bloques en dos filas distintas de un mismo banco.	☐SÍ	☐NO	**(NO)**
c) Dos bloques en dos bancos distintos.	☐SÍ	☐NO	**(SÍ)**

4 ***(1 punto)***

A) Considera un MIPS al que se ha conectado un módulo de memoria M1 de 256 MB en la dirección más alta del mapa (esto es, que contiene la dirección 0xFFFFFFFF)

A.1 ¿Cuál es la menor dirección de las contenidas en el módulo?

0xF0000000

A.2 ¿Cuál será la función de selección activa por nivel bajo?

/A31 + /A30 + /A29 + /A28

B) Disponéis de un módulo M2 de 128 MB y queréis instalarlo de manera que no quede espacio libre entre este y M1.

B.1 ¿Cuál es la menor dirección de las contenidas en el módulo?

0xE8000000

B.2 ¿Cuál será la función de selección activa por nivel bajo?

/A31 + /A30 + /A29 + A28 + /A27

C) Un día encontráis instalado otro módulo M3 de memoria con la función de selección activa por nivel bajo

SelM3 = A31 + /A30 + A29

C.1 ¿Cuál es la menor dirección de las contenidas en el módulo?

0x40000000

C.2 ¿Cuál es el tamaño del espacio libre comprendido entre esta dirección y la dirección 0x00000000? Exprésalo en MB

400h = 1024 MB

5 ***(2 puntos)*** Se dispone de una memoria cache de datos de 32 KB y memoria principal de 4 GB. Por limitaciones de diseño, el tamaño de la memoria de control asociada a esta cache no debe sobrepasar el 10% del tamaño neto para datos. Indica cuáles de las siguientes configuraciones de cache cumplen esta restricción y cuáles no. Justifica las respuestas indicando el número de entradas de la memoria de control y el formato de cada entrada en cada uno de los casos.

Tamaño bloque (Bytes)	Correspondencia	Política escritura (acierto)	Política escritura (fallo)	Algoritmo de reemplazo	¿Cumple?
16	Directa	Write through	No allocate	—	NO
16	4 vías	Write back	Allocate	LRU	NO
32	Directa	Write through	No allocate	—	SÍ
32	4 vías	Write back	Allocate	LRU	SÍ

La restricción del 10% supone que la memoria de control no debe superar el tamaño de 32 KB / 10 = 3276,8 Bytes (en la práctica, podemos truncar a 3276 B).

Caso 1: Hay 32 KB / 16 B = 2048 líneas. Las direcciones de memoria (de 32 bits, ya que el espacio direccionable total es de 4 GB) se estructuran en 4 bits para offset, 11 bits para etiqueta y 17 bits de etiqueta. Los bits de control necesarios para esta configuración se limitan a 1 (V) + 17 (Etiq) = 18 por cada línea, es decir: 18 * 1024 = 36864 b = 4608 B, que es mayor que el 10 % de 32 KB.

Caso 2: Ahora tenemos 2048 / 4 = 512 conjuntos, luego hay 19 bits de etiqueta, 9 de conjunto y 4 de offset. Para el control hacen falta 1 (V) + 19 (Etiq) + 1 (Dirty) + 2 (LRU) = 23 bits por línea, es decir: 23 * 2048 = 47104 b = 5888 B, por lo que tampoco cumple.

Caso 3: Ahora tenemos 32 KB / 32 B = 1024 líneas. Las direcciones tienen 5 bits de offset, 10 bits de línea y 17 de etiqueta. La memoria de control requiere 1 (V) + 17 (Etiq) = 18 bits, que por el número de líneas serán 18432 b, es decir, 2304 B. Esta configuración sí cumple la restricción.

Caso 4: Habrán 1024 líneas / 4 = 256 conjuntos. Direcciones de memoria: 5 bits offset, 8 para conjunto y los 19 restantes para etiqueta. La memoria de control requiere: 1 (V) + 19 (Etiq) + 1 (Dirty) + 2 (LRU) = 23 bits. Multiplicando por el número de líneas tendremos un tamaño total de 23 * 1024 = 23552 b = 2944 B, por lo que esta configuración también cumple la restricción.

Resumen:

Caso	Estructura dirección E+C+D	Bits control/línea V+E+D+LRU	Num. de líneas	Total control en bits	en bytes
1	17+11+4	1+17 = 18	2048	36864	4608
2	19+9+4	1+19+1+2 = 23	2048	47104	5888
3	17+10+5	1+17+0+0 = 18	1024	18432	2304
4	19+8+5	1+19+1+2 = 23	1024	23552	2944

6 ***(2.5 puntos)*** Considérese el programa que hace los cálculos siguientes sobre un vector V de N componentes:

a) Calcula la suma de las componentes del vector $f0 = \sum_{i=0}^{N-1} V[i]$

b) Hace la división de todas las componentes del vector por el valor obtenido en a)

Seguidamente se muestra el código correspondiente en un lenguaje de alto nivel y en ensamblador del MIPS R2000. Obsérvese que se hacen dos recorridos del vector, ambos en orden creciente de direcciones. *N* es una constante que toma el valor indicado más abajo. Cada pseudoinstrucción que aparece en el código fuente de ensamblador se traduce en una única instrucción máquina.

Alto nivel	Ensamblador
`float V[N];`	`.data 0x10000000`
`float f0;`	`V: .float 2.0,... # N componentes`
`int t0,t1;`	
	`.text 0x00400000`
`f0 = 0.0; t0=0;`	`la $t0,V`
`for (t1=N; t1>0; t1--){`	`li $t1,N`
`  f0 = f0 + V[t0];`	`mtc1 $zero,$f0`
`  t0 = t0 + 1;`	`for1: lwc1 $f10,0($t0)`
`}`	`add.s $f0,$f0,$f10`
`t0=0;`	`addi $t1,$t1,-1`
`for (t1=N; t1>0; t1--){`	`addi $t0,$t0,4`
`  V[t0] = V[t0]/f0;`	`bgtz $t1,for1`
`  t0 = t0 + 1;`	
`}`	`la $t0,V`
	`li $t1,N`
	`for2: lwc1 $f10,0($t0)`
	`div.s $f10,$f10,$f0`
	`swc1 $f10,0($t0)`
	`addi $t1,$t1,-1`
	`addi $t0,$t0,4`
	`bgtz $t1,for2`

El procesador dispone de memorias cache separadas para instrucciones y para datos (1KB + 1KB) con bloques de 32 bytes. En el momento en que comienza la ejecución del programa, todas las líneas de la cache son inválidas.

A) ¿Cuántas instrucciones caben en un bloque? ¿Cuántas componentes del vector V caben en un bloque?

Instrucciones / bloque de código: 8	Componentes de V / bloque de datos: 8

B) Supón que las memorias cache son de correspondencia directa, escritura posterior (*write back*) y sin ubicación en escritura (*no-write-allocate*). Con N=100, calcula los valores siguientes al final de la ejecución del código:

Memoria cache	Parámetro	Valor
Instrucciones	Número de accesos	5 + 11xN = 1105
	Número de bloques referenciados	15 instr. x 4 bytes/inst / 32 bytes/bloque = 2 bloques
	Tasa de aciertos	1 – 2/1105 ≈ 100 %
Datos	Número de accesos	3 x 100 = 300
	Número de bloques referenciados	N x 4 / 32 = 13
	Número de reemplazos	0
	Tasa de aciertos	1 – 13/300= 0.957
	Número de escrituras de bloque en memoria principal	0

C) Si N = 260, con una memoria cache de las mismas características, calcula:

Memoria cache	Parámetro	Valor
Datos	Número de accesos	3 x 260 = 780
	Número de bloques referenciados	260 x 4 / 32 = 33
	Número de reemplazos	3
	Tasa de aciertos	1 – 35/780= 0.955
	Número de escrituras de bloque en la memoria principal	1
	Número de bloques de cache no coherentes con memoria	32

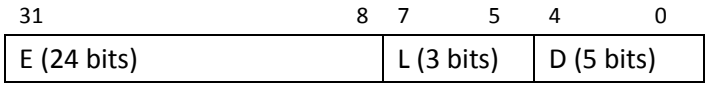
D) ¿Qué cambiaría en el apartado B (con N=100) si la memoria cache de datos aplicara las políticas de escritura directa *(write-through)* y con ubicación *(write-allocate)*

Cada una de las escrituras **`swc1 $f10,0($t0)`** sigue a una lectura **`lwc1 $f10,0($t0)`** en la misma dirección. Por lo tanto, no hay ningún fallo de escritura y la política de ubicación es irrelevante. Es decir: todas las escrituras son aciertos y el cambio no afecta a la tasa de fallos.

Con la política *write-through*, las escrituras de una palabra en un bloque de la cache se transmiten directamente a la memòria principal, y nunca hay incoherencia entre ambos niveles de la jerarquía de memoria.

Si N=100, cada una de las 100 escrituras en la memoria cache hubiera provocado la escritura de la palabra implicada en la memoria principal. Al final del programa los 13 bloques de datos que quedan en la cache serían coherentes con la memoria principal.

7 ***(1.5 puntos)*** Un procesador semejante al MIPS tiene conectada una memoria cache de datos de correspondencia directa y escritura posterior (*write-back*) con ubicación (*write-allocate*) formada por 8 líneas que contienen bloques de 32 bytes. La estructura de la dirección que interpreta la memoria cache es



La tabla de la derecha muestra el estado inicial de la memoria cache

Línea	V	M	E (hex)
0	1	1	**100000**
1	0	–	**------**
2	1	0	**000200**
3	1	0	**1af002**
4	0	–	**------**
5	0	–	**------**
6	1	0	**87654f**
7	1	1	**000666**

A) ¿Cuál es el rango de direcciones del bloque contenido en la línea 6?

De 0x87654fC0 a 0x87654fdf

B) Explica cómo afecta al estado de la cache cada una de las instrucciones que tenéis más abajo. En cada caso tienes de contestar cuál es la línea afectada, si se trata de un caso de acierto (A) o de fallo (F) y el estado en que quedan los bits de válido (V), modificado (M) y la etiqueta (E). Considera que **$t0 = 0x10000000**. Debéis partir siempre del estado inicial.

			Estado resultante		
	Línea	A/F	V	M	E
`lw $t1,0($t0)`	**0**	**A**	**1**	**1**	**1000000**
`lw $t1,0x120($t0)`	**1**	**F**	**1**	**0**	**1000001**
`sw $t1,0x120($t0)`	**1**	**F***	**1**	**1**	**1000001**

(*) En algunos casos se ha interpretado que el `sw` seguía en secuencia al `lw`, por lo que debería ser A