ESTRUCTURA DE COMPUTADORES

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

Apellidos y nombre	DNI	Grupo

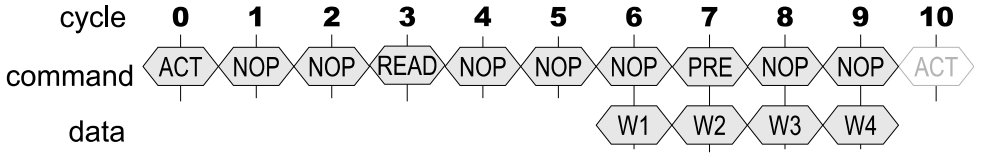
1 ***(1 punto)*** Una CPU tiene una capacidad de direccionamiento de 16 GB y emplea palabras de 64 bits.

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

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

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

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.

b) Cuál es su **ancho de banda**.

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

b) Dos bloques en dos filas distintas de un mismo banco. ☐SÍ ☐NO

c) Dos bloques en dos bancos distintos. ☐SÍ ☐NO

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?

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

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?

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

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?

C.2 ¿Cuál es el tamaño del espacio libre comprendido entre esta dirección y la dirección 0x00000000? Exprésalo en 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	—	
16	4 vías	Write back	Allocate	LRU	
32	Directa	Write through	No allocate	—	
32	4 vías	Write back	Allocate	LRU	

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?

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:

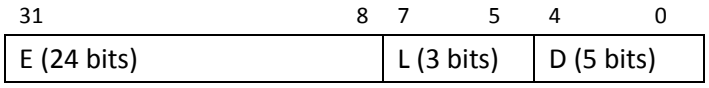
Memoria cache	Parámetro	Valor
Instrucciones	Número de accesos	
	Número de bloques referenciados	
	Tasa de aciertos	
Datos	Número de accesos	
	Número de bloques referenciados	
	Número de reemplazos	
	Tasa de aciertos	
	Número de escrituras de bloque en memoria principal	

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

Memoria cache	Parámetro	Valor
Datos	Número de accesos	
	Número de bloques referenciados	
	Número de reemplazos	
	Tasa de aciertos	
	Número de escrituras de bloque en la memoria principal	
	Número de bloques de cache no coherentes con memoria	

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)*

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?

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)`					
`lw $t1,0x120($t0)`					
`sw $t1,0x120($t0)`					