grapa

Examen 2015/16-1

Asignatura	Código	Fecha	Hora inicio
Fundamentos de computadores	75.562	13/01/2016	09:00

⊂75.562ℜ13ℜ01ℜ16ℜEΞ&∈
75.562 13 01 16 EX

Espacio para la etiqueta identificativa con el código personal del **estudiante**.
Examen

Este enunciado corresponde también a las siguientes asignaturas:

- 81.518 - Fundamentos de computadores

Ficha técnica del examen

- **Comprueba que el código y el nombre de la asignatura corresponden a la asignatura de la cual estás matriculado.**
- **Debes pegar una sola etiqueta de estudiante en el espacio de esta hoja destinado a ello.**
- **No se puede añadir hojas adicionales.**
- **No se puede realizar las pruebas a lápiz o rotulador.**
- **Tiempo total** 2 **horas**
- **En el caso de que los estudiantes puedan consultar algún material durante el examen, ¿cuál o cuáles pueden consultar?:** No se puede consultar ningún material.
- **Valor de cada pregunta:** Prob.1: 20%, Prob. 2: 35%, Prob. 3: 35%, Prob. 4: 10%.
- **En el caso de que haya preguntas tipo test: ¿descuentan las respuestas erróneas?** NO **¿Cuánto?**
- **Indicaciones específicas para la realización de este examen**
  - No se puede utilizar ningún tipo de calculadora.
  - Razonad las respuestas en cada ejercicio. Las respuestas sin justificar no recibirán puntuación.

Enunciados

Examen 2015/16-1

Asignatura	Código	Fecha	Hora inicio
Fundamentos de computadores	75.562	13/01/2016	09:00

PROBLEMA 1 [20%]

a) **[5%]** Dada la secuencia de bits 10000110, que representa un número entero codificado en complemento a 2 y en 8 bits, indicad cuál es el valor de este número en decimal.

Dado que el bit de más peso es 1, sabemos que se trata de un número negativo. Podemos encontrar su valor en decimal aplicando el TFN, pero dando signo negativo al primer sumando:

$$-1\cdot2^7 + 0\cdot2^6 + 0\cdot2^5 + 0\cdot2^4 + 0\cdot2^3 + 1\cdot2^2 + 1\cdot2^1 + 0\cdot2^0 = -128 + 4 + 2 = -122_{(10}.$$

También lo podríamos hacer cambiando de signo el número e interpretando el resultado como número natural, con lo cual hallaríamos su valor absoluto:

Intercambiamos 0s y 1s: 10000110 -> 01111001
Sumamos 1 al resultado: 01111001 + 1 = 01111010

$$0\cdot2^7 + 1\cdot2^6 + 1\cdot2^5 + 1\cdot2^4 + 1\cdot2^3 + 0\cdot2^2 + 1\cdot2^1 + 0\cdot2^0 = 64 + 32 + 16 + 8 + 2 = 122_{(10}.$$

b) **[5%]** Dado el número real $23{,}23_{(10}$, indicad cuál es la codificación de este número en un formato de signo y magnitud en coma fija con 1 bit para el signo, 5 bits para la parte entera y 4 bits para la parte fraccionaria.

Es un número positivo, y por lo tanto el bit de signo valdrá 0.
Encontramos la representación de la parte entera por el método de la división entera:

$$\begin{aligned} 23 &= 11\cdot2 + \mathbf{1} \\ 11 &= 5\cdot2 + \mathbf{1} \\ 5 &= 2\cdot2 + \mathbf{1} \\ 2 &= 1\cdot2 + \mathbf{0} \\ 1 &= 0\cdot2 + \mathbf{1} \end{aligned}$$

La representación de la parte entera es 10111.
Encontramos la representación de la parte fraccionaria haciendo multiplicaciones sucesivas:

$$\begin{aligned} 0{,}23\cdot2 &= 0{,}46 + \mathbf{0} \\ 0{,}46\cdot2 &= 0{,}92 + \mathbf{0} \\ 0{,}92\cdot2 &= 0{,}84 + \mathbf{1} \\ 0{,}84\cdot2 &= 0{,}68 + \mathbf{1} \end{aligned}$$

Detenemos el cálculo aquí, porque ya hemos obtenido 4 bits de la parte fraccionaria, que son los que podemos usar en esta representación, y no hemos llegado a un resultado fraccionario 0.

Si juntamos todas las partes obtenemos que, en el formato de representación que nos han dado, el número $23{,}23_{(10}$ se representa $0101110011_{(SM2}$.

c) **[10%]** Dados los números $A = 10011101_{(SM2}$ y $B = 11010011_{(SM2}$, codificados en 8 bits en signo y magnitud, calculad el resultado de la operación aritmética *A* + *B*, indicando si se produce desbordamiento en el cálculo.

Los dos números son negativos. Por lo tanto, sumaremos sus magnitudes y daremos el mismo signo al resultado.

Examen 2015/16-1

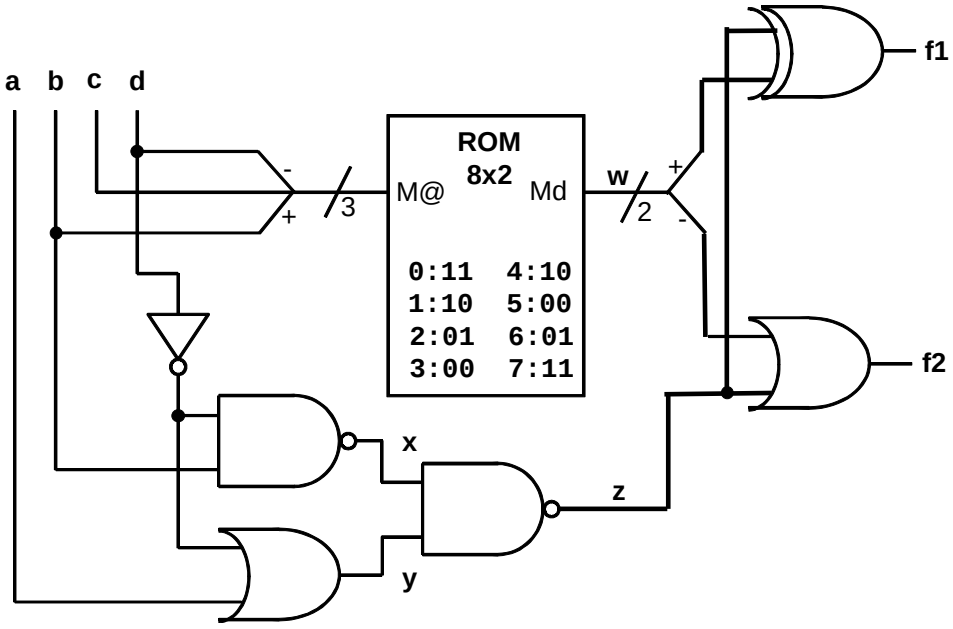
Asignatura	Código	Fecha	Hora inicio
Fundamentos de computadores	75.562	13/01/2016	09:00

```
    1  1  1  1  1      ← acarreo
 0  0  1  1  1  0  1   ← |A|
-1  0  1  0  0  1  1   ← |B|
 1  1  1  0  0  0  0
```

No se produce acarreo en la última etapa de la suma, y por lo tanto no hay desbordamiento. Añadiendo el bit de signo obtenemos que A + B = $11110000_{(SM2}$.

PROBLEMA 2 [35%]

a) **[20%]** Dado el circuito lógico combinacional siguiente:



donde el contenido de la memoria ROM viene especificado en el mismo bloque con la forma **dirección:contenido**, donde la dirección se expresa en decimal y el contenido de la memoria se expresa en binario.

Se pide que rellenéis la siguiente tabla que determina los valores de las funciones intermedias especificadas y de las de salida del circuito anterior para unas determinadas combinaciones de las variables de entrada.

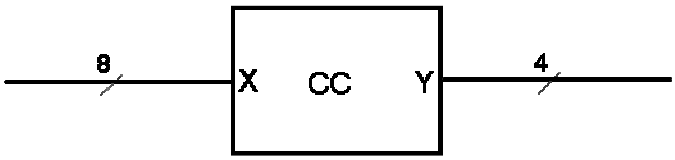
a	b	c	d	w_1	w_0	x	y	z	f1	f2
0	0	0	0							
0	1	0	1							
1	0	1	1							
1	1	0	0							

a	b	c	d	w_1	w_0	x	y	z	f1	f2
0	0	0	0	1	1	1	1	0	1	1
0	1	0	1	0	0	1	0	1	1	1
1	0	1	1	0	0	1	1	0	0	0
1	1	0	0	1	0	0	1	1	0	1

Examen 2015/16-1

Asignatura	Código	Fecha	Hora inicio
Fundamentos de computadores	75.562	13/01/2016	09:00

b) **[15%]** Diseñad el circuito combinacional *CC* que sirve para obtener información de cara a normalizar una mantisa.



La entrada del circuito *X* corresponde a la magnitud de un número real codificado en coma fija y en 8 bits, donde hay 3 bits (los de más peso) para representar la parte entera y los 5 bits restantes representan la parte fraccionaria. El número en la entrada nunca podrá ser 0.

La salida *Y* corresponde a un número entero representado en complemento a 2 y en 4 bits que especifica el número de bits que hay que desplazar la coma del número *X* para obtener una mantisa normalizada (la coma tiene que estar a la derecha del primer bit significativo). Si la coma se tiene que desplazar a la izquierda de la mantisa inicial *X* entonces el signo de *Y* será positivo, y si la coma se tiene que desplazar a la derecha entonces el signo de *Y* será negativo.

Ejemplos:

- Si *X*=10101010 entonces *Y*=+2 (puesto que inicialmente tenemos 101,01010 y la mantisa normalizada es 1,0101010 de forma que la coma se ha desplazado 2 posiciones a la izquierda).

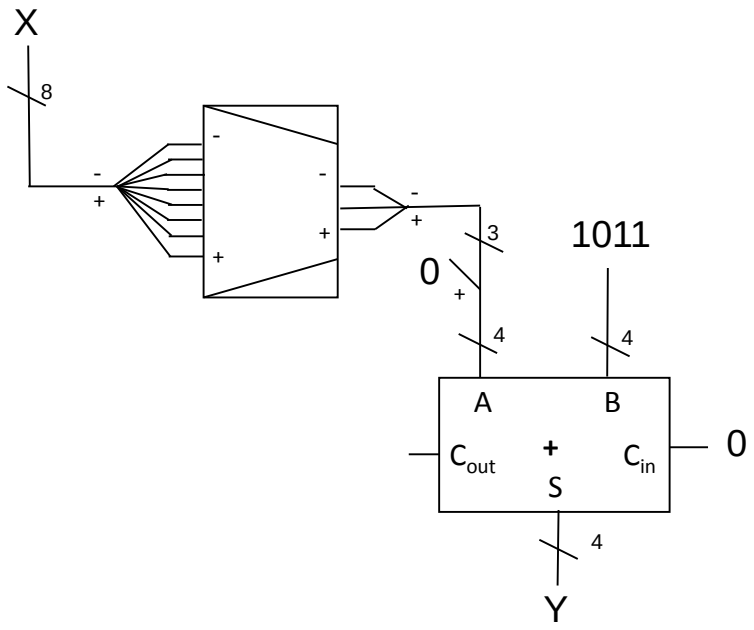
- Si *X*=00000010 entonces *Y*=-4 (mantisa inicial: 0,00010; mantisa normalizada: 1,0)

Podéis usar los bloques y puertas que consideráis necesarios, excepto memorias ROM.

Para normalizar la mantisa tenemos que saber en qué posición se encuentra en la entrada *X* el bit con valor 1 de más peso. La diferencia entre esta posición y la posición de la coma en el formato dado corresponde a cuántos bits se tiene que mover la coma para obtener una mantisa normalizada. Obtenemos la posición del bit con valor 1 de más peso de *X* con un codificador. Por lo tanto, la salida del codificador indica esta posición. Para obtener la diferencia con la posición de la coma (la parte fraccionaria son los 5 bits de menor peso de *X*), tenemos que restarle 5. Convertimos esta resta en una suma con el valor -5, donde representamos -5 en Ca2, que corresponde a $1011_{(Ca2}$. Usamos un sumador donde la salida del sumador corresponde a la salida *Y* del circuito que indica en Ca2 el número de bits que hay que desplazar la coma.

Examen 2015/16-1

Asignatura	Código	Fecha	Hora inicio
Fundamentos de computadores	75.562	13/01/2016	09:00

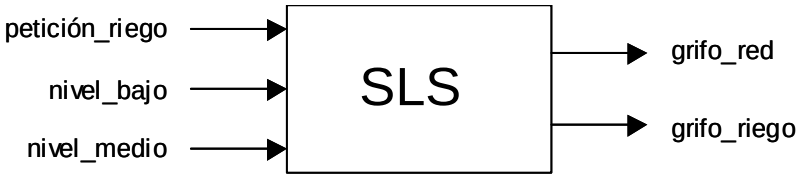


PROBLEMA 3 [35%]

a) **[20%]** Se quiere diseñar un circuito que controle el funcionamiento de una cisterna de agua para regar una huerta. La cisterna se llena de forma natural con el agua de la lluvia, y también se puede llenar con agua de la red. La cisterna tiene los siguientes elementos:

- Un grifo de entrada de agua de la red, que sólo está abierto mientras la señal *grifo_red* vale 1.
- Un grifo de salida de agua para regar, que sólo está abierto mientras la señal *grifo_riego* vale 1.
- Dos sensores de nivel del agua, que generan estas señales:
  - *nivel_bajo*: vale 1 sólo si en la cisterna hay menos de 200 litros.
  - *nivel_medio*: vale 1 sólo si en la cisterna hay 10.000 litros o más.
- La cisterna puede tener alguna fisura, por la cual se puede perder agua.

El circuito que se quiere diseñar tiene las señales de entrada y salida siguientes:



La señal *petición_riego* vale 1 cuando desde la huerta se pide agua para regar. El resto de señales son los descritos anteriormente.

El funcionamiento tiene que ser el siguiente:

Examen 2015/16-1

Asignatura	**Código**	**Fecha**	**Hora inicio**
Fundamentos de computadores	75.562	13/01/2016	09:00

- Si hay una demanda de petición de agua desde la huerta, se servirá siempre que en la cisterna haya como mínimo 200 litros de agua. Servir una petición quiere decir dar agua para regar durante 3 ciclos de reloj.
- Mientras se está sirviendo una petición de riego, no se hará caso de otra petición que pudiera llegar.
- Si llegan a la vez una petición de riego y el aviso de nivel bajo, se dará prioridad a atender el aviso de nivel bajo.
- Cuando en la cisterna haya menos de 200 litros de agua:
  - No se servirá ninguna nueva petición de riego.
  - Si se estaba sirviendo una, se interrumpirá. Cuando el nivel de agua en la cisterna llegue o supere los 10.000 litros, se dará al regante la cantidad de agua que le faltaba para completar el servicio.
  - Se abrirá el grifo de agua de la red, y se mantendrá abierto hasta que en la cisterna haya al menos 10.000 litros.

Dibujad el grafo de estados que describe el comportamiento del circuito. Suponed que en el estado inicial no hay ningún grifo abierto, y el nivel de agua está entre 200 y 10.000 litros.

Denominaremos INI al estado inicial. Estando en este estado, si llega una petición de riego y hay agua suficiente pasaremos a un estado en el que se inicia el servicio de riego. Tendrá que haber dos estados más en los que se riegue, puesto que cada servicio dura tres ciclos. Denominaremos a estos estados RIEGO1, RIEGO2 y RIEGO3.

En cambio, si estando en INI el nivel de agua cae por debajo de 200 litros habrá que ir a un estado para llenar la cisterna (abriendo el grifo de la red y cerrando el de riego), independientemente de si llega una petición, puesto que llenar la cisterna es prioritario ante servir un riego. Denominaremos a este estado LLENAR. Habrá que permanecer en este estado hasta que *nivel_medio* = 1, y entonces volveremos al estado inicial si no llega ninguna petición, o empezaremos un servicio de riego si llega una.

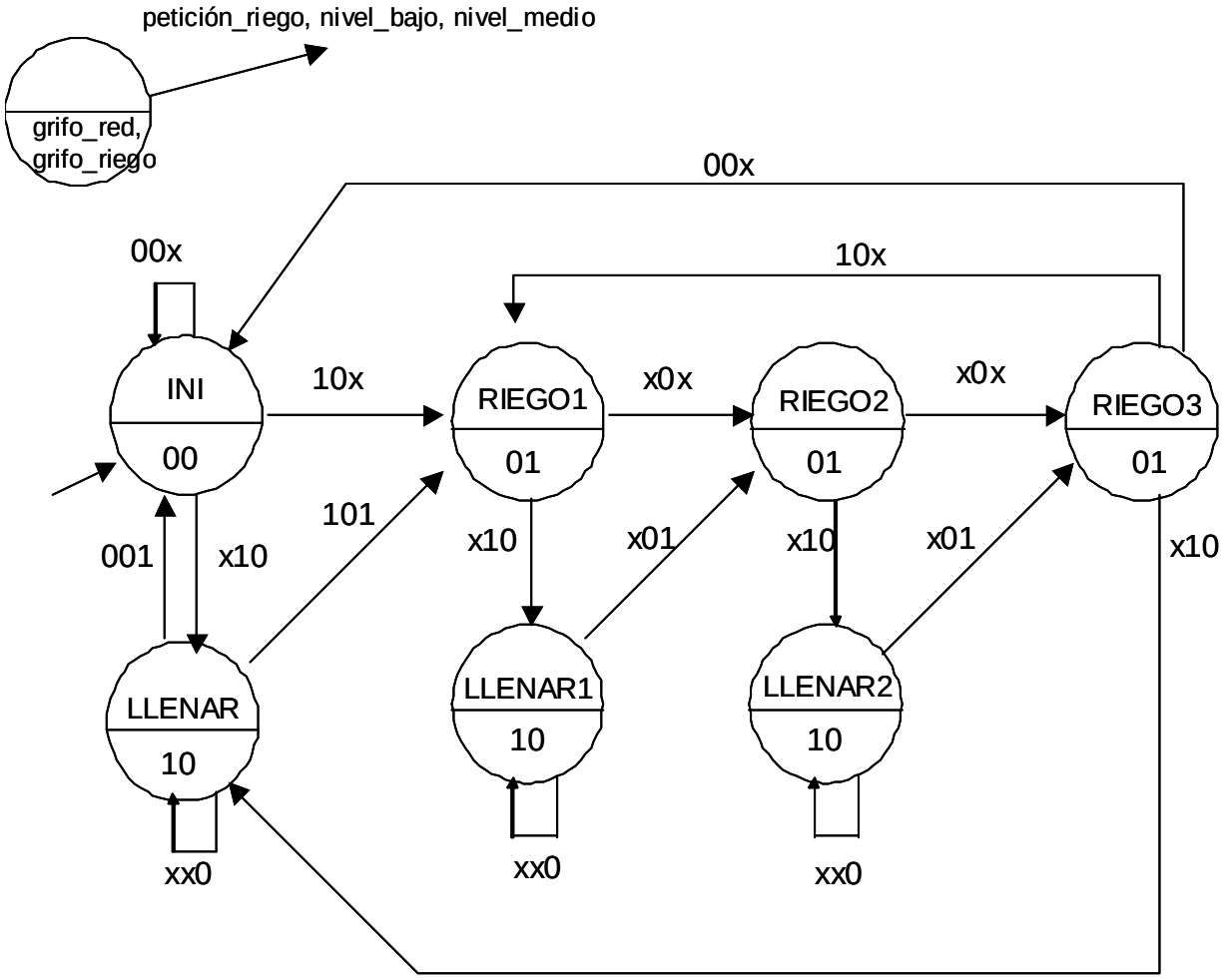
Si durante un servicio el nivel del agua baja por debajo de 200 litros tendremos que pasar a un estado en el que se llene la cisterna; pero ese estado no puede ser LLENAR, porque ahora hay que recordar que ya hemos servido el primer ciclo del riego. Denominaremos LLENAR1 al nuevo estado. Cuando en la cisterna ya haya 10.000 litros, retomaremos el servicio del riego, pasando al estado RIEGO2.

Análogamente hará falta un estado LLENAR2, que recuerde que ya hemos servido dos ciclos del riego. En cambio no hará falta lo que sería un estado LLENAR3, porque una vez completado un servicio de riego estamos en la misma situación que en el estado inicial (que es que no hay ningún riego servido a medias), y por lo tanto las transiciones que salen de RIEGO3 irán a los mismos estados que las que salen del estado INI.

El grafo completo es el siguiente. En él no figuran las combinaciones imposibles de las variables de entrada, es decir aquellas en las que las señales *nivel_bajo* y *nivel_medio* valen 1 a la vez.

Examen 2015/16-1

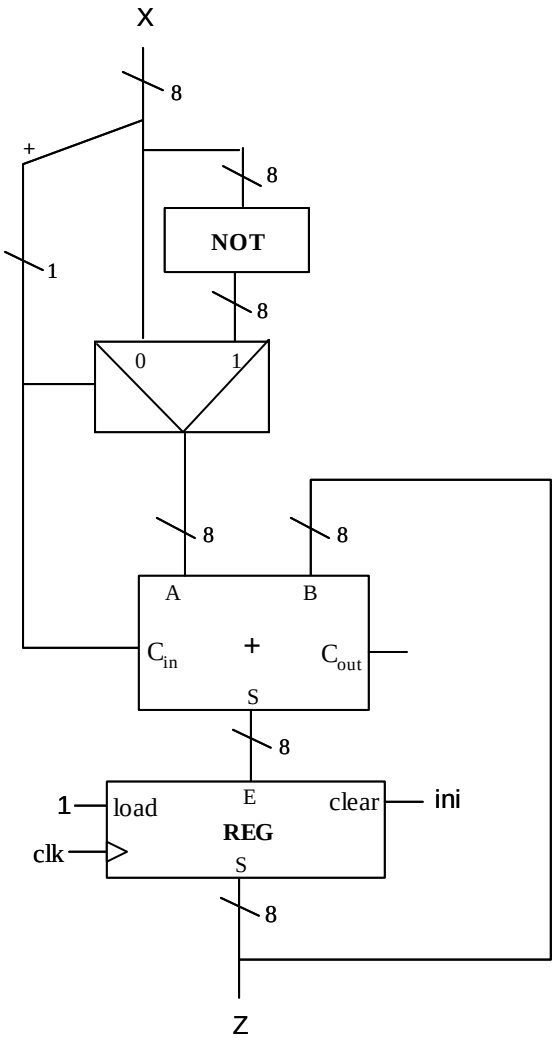
Asignatura	**Código**	**Fecha**	**Hora inicio**
Fundamentos de computadores	75.562	13/01/2016	09:00



b) **[10%]** Tenemos el circuito de la figura, en el que la entrada *X* y la salida *Y* son números enteros representados en Ca2. Suponed que la señal *ini* vale 1 durante cierto tiempo, después pasa a 0 y no vuelve a cambiar de valor.

Examen 2015/16-1

Asignatura	Código	Fecha	Hora inicio
Fundamentos de computadores	75.562	13/01/2016	09:00



Explicad con palabras el significado del valor que se guarda en REG a partir del momento en que *ini* pasa a 0, escribiendo el razonamiento que seguís para llegar a la respuesta.

El registro REG inicialmente vale 0 (puesto que a su entrada *clear* hay conectado *ini*), y después de que *ini* pase a 0 se carga a cada ciclo (*load* = 1) con la salida de un sumador, al cual llegan por un lado REG y por el otro un valor que depende de la entrada *X*.

En la entrada *X*, representada en Ca2, puede haber un número positivo o negativo. Si es negativo, su bit de más peso es 1, y en este caso el multiplexor selecciona el complemento de *X*, que se obtiene con el bloque NOT. El bit de más peso de *X* también está conectado a la entrada C_{in} del sumador. Por lo tanto, si *X* es negativo el sumador calculará REG + *X*' + 1 = REG – X.

En cambio, si *X* es positivo el cálculo será REG + X.

Por lo tanto, la salida del sumador siempre valdrá REG + |*X*|.

Vemos pues que REG guarda la suma acumulada de los valores absolutos de *X*.

Asignatura	Código	Fecha	Hora inicio
Fundamentos de computadores	75.562	13/01/2016	09:00

c) **[5%]** En el circuito anterior, ¿cómo podemos saber si el valor de REG es correcto, es decir, si se ha producido desbordamiento en el cálculo?
El registro tiene que tener siempre valores positivos. Por lo tanto, su bit de signo, que corresponde a su bit de más peso, tiene que valer 0. Si el bit de más peso es 1 (indica valor negativo), se habrá producido desbordamiento.

PROBLEMA 4 [10%]

a) **[5%]** ¿Qué es la arquitectura de Von Neumann?

Una manera de construir máquinas que tienen una memoria común para las instrucciones y los datos.

b) **[5%]** ¿Para qué se usa la memoria caché?

Para proporcionar a la CPU un acceso más rápido a la información de la memoria principal.