APELLIDOS	NOMBRE	GRADO
	SOLUCIÓN	

Ejercicio 1 (40 minutos – 4 puntos)

La siguiente figura muestra un diagrama de bloques simplificado de la arquitectura interna de un microprocesador:

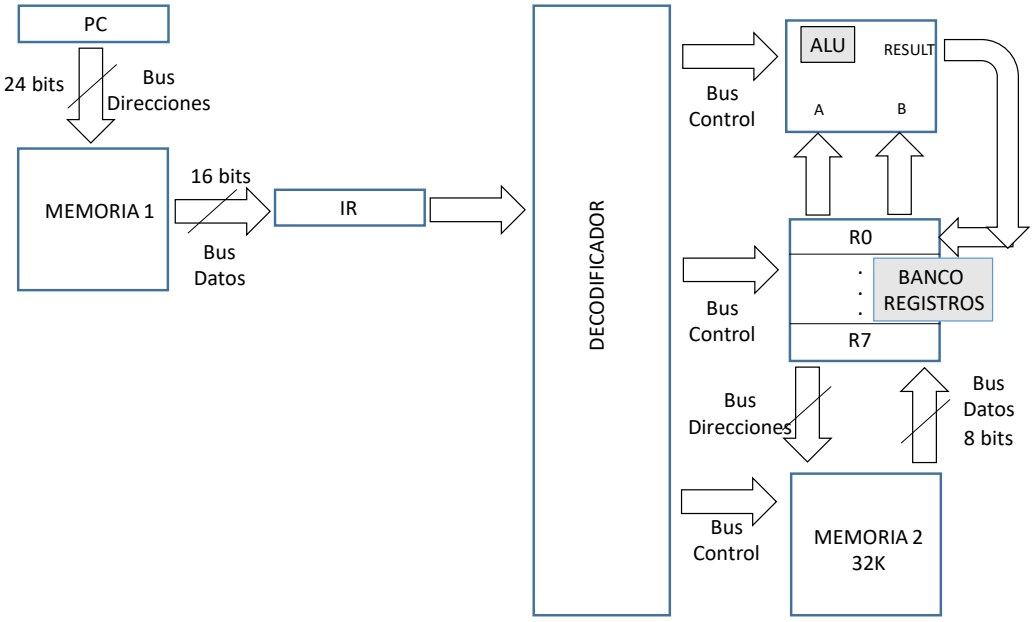


Figura 1

Responde a las siguientes preguntas de acuerdo a la información proporcionada en la figura. Justifica sus respuestas:

1- ¿Es una arquitectura Harvard? *(1 punto)*

Es una arquitectura Harvard, ya que tiene totalmente separadas dos memorias, una para programa (Memoria 1) y otra para datos (Memoria 2). Los registros internos sólo tienen acceso a la Memoria 2, y nunca a la Memoria 1.

2- Indique la anchura del bus de datos y del bus de direcciones, así como el tamaño total de la memoria/s del sistema. Especifique en su respuesta si la memoria/memorias son de datos, de código o de ambos tipos. *(1 punto)*

Para la memoria de programa (Memoria 1), los buses necesarios son uno de datos de 16 bits, y uno de direcciones de 24 bits, tal y como se indica en la figura. Con esos tamaños de buses, Memoria 1 podría llegar a tener un tamaño de 16M x 16.

Para el caso de la memoria de datos (Memoria 2), la figura nos indica que el bus de datos para la memoria de datos es de 8 bits, pero no indica nada relativo al bus de direcciones. Si consideramos que la Memoria 2 expuesta tiene la mayor capacidad posible, entonces el bus de direcciones de dicha memoria es de 15 bits. Y por tanto el tamaño máximo de la Memoria 2, sería de 32K x 8.

3- Tamaño del IR y del PC *(1 punto)*

Con los datos dados en la figura, el PC tendrá 24 bits, mientras que el IR tendrá 16 bits, que corresponden, respectivamente, al tamaño del bus de direcciones de la memoria de programa, y al del bus de datos de la memoria de programa.

4- Explica en unas pocas líneas qué es y cómo se hace la fase fetch de la unidad de control. *(1 punto)*

La fase de fetch en una unidad de control corresponde a la captura de la instrucción a ejecutar, antes de ser decodificada y ejecutada. Por lo tanto, en dicha fase, lo que se hace es:

MAR <- PC
PC <- PC + 1
MBR <- (MAR)
IR <- MBR

5- ¿Para qué se utiliza el PC? *(1 punto)*

El PC es el registro que contiene la dirección de la siguiente instrucción a ejecutar.

6- ¿Cuantos bits se necesitan para direccionar un registro del banco de registros internos dentro de una instrucción? *(1 punto)*

En la Figura se muestran sólo 8 registros (R0 – R7), por lo que se necesitarían 3 bits para codificar el número de registro dentro de una instrucción.

7- Explique qué se entiende por modo de direccionamiento. Proporcione dos ejemplos de dos modos de direccionamiento típicos, y cómo afectan a la codificación de la instrucción *(2 puntos)*

Modo de direccionamiento es la forma en la que se indica en una instrucción cómo obtener la dirección efectiva de un operando. Dos ejemplos de modos de direccionamiento son el inmediato y el indexado. El inmediato implica que el dato del operando está directamente codificado en la instrucción, lo que implica que hay que dejar espacio en la instrucción para codificar el dato (en este ejemplo, 8 bits). El indexado es aquel en el que la dirección de memoria donde se encuentra el dato del operando, mediante la suma del contenido de un registro, que contiene la dirección base, y otro registro, que contiene el índice variable. Por lo tanto, en la codificación de la instrucción hay que reservar espacio para dos registros, que en el caso de este ejemplo sería 3 bits para Rb y otros 3 bits para Ri.

8- Organización de la memoria: *(2 puntos)*

a) Tenemos varios chips que implementan la memoria etiquetada como 1 en la Figura:
- 4 chips de 4Mx8 bits
- 1 chip de 4Mx16 bits
- 1 chip de 8Mx16 bits

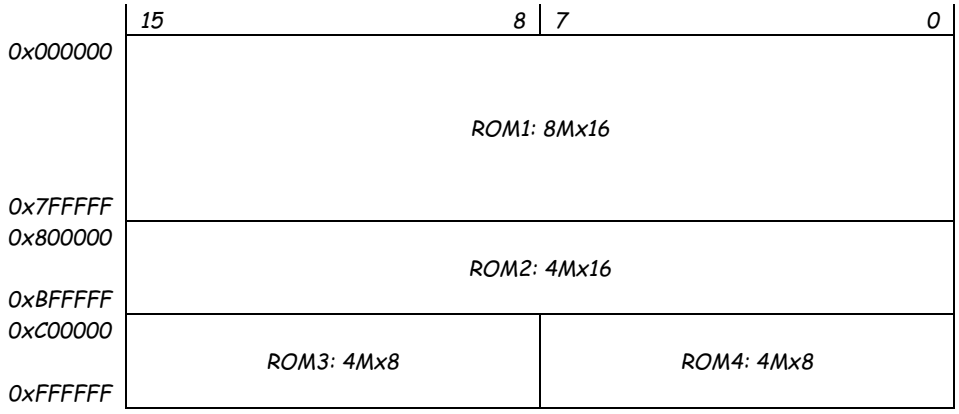
Usa el mínimo número de ellos para implementar la memoria y dibuja el mapa de memoria. El chip más grande debe estar en las direcciones más bajas. Especifica los rangos de direcciones asignados a cada chip en hexadecimal.

Use un decodificador 2:4 y puertas lógicas, para obtener las señales CS se cada chip de memoria. Las señales CS son activas a nivel bajo.

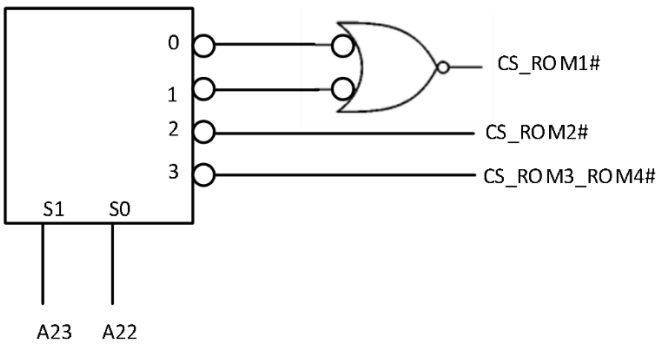
Tal y como se ha comentado antes, si la Memoria 1 tiene la máxima capacidad permitida, debe ser de 16M x 16. Por lo tanto, su implementación con un número mínimo de chips sería utilizar:

- *1 chip de 8Mx16 + 1 chip de 4Mx16 + 2chips de 4Mx8*

El mapa de memoria sería:



El decodificador sería:



Ejercicio 2 (20 minutos – 3 puntos)

Se tiene una arquitectura Von Neumann de 16 bits, con capacidad de direccionar 128MB, sólo a nivel de palabra. Dicha arquitectura contiene 8 registros internos propósito general y filosofía Load & Store. Con esta información, conteste a las siguientes preguntas:

1) (70%) Diseñe una codificación de los distintos tipos de instrucciones microprocesador, minimizando en lo posible el tamaño de la instrucción ajustando a números enteros de palabras (es decir, las instrucciones pueden ocupar una palabra o más de una; en el caso de más de una, la primera que se leerá contendrá, al menos, el opcode). Los tipos de instrucciones que debe tener el microprocesador, son:

- 2 instrucciones de transferencia de datos entre memoria y registros internos, con direccionamiento directo.
- 5 instrucciones de transferencia de datos entre memoria y registros internos, con direccionamiento indexado.
- 2 instrucciones de transferencia de datos entre memoria y registros internos, con direccionamiento indirecto con desplazamiento, dando el desplazamiento en complemento a 2, con 10 bits.
- 14 instrucciones aritmético/lógicas de operar entre registros, siendo el resultado el mismo registro que uno de los operandos.
- 3 instrucciones aritmético/lógicas de operar entre registros, con uno de los operandos dado por direccionamiento inmediato, limitado a 8 bits, y dando el resultado en el otro operando.
- 9 instrucciones de control con direccionamiento inherente.
- 7 saltos condicionales con direccionamiento relativo a contador de programa, siendo el desplazamiento relativo de más/menos 1Kpalabra.

2) (30%) Indique una respuesta justificada a cada una de las siguientes preguntas:

a) Número máximo de palabras que ocupa una instrucción
b) Número mínimo de palabras que ocupa una instrucción
c) Tamaño del Registro de Instrucción
d) Tamaño del Contador de Programa
e) Tamaño del Puntero de Pila
f) Tamaño de los Registros internos

1) Se trata de una arquitectura Von Neumann de 16 bits (es decir, 16 bits de datos), y capaz de direccionar 128MB, pero sólo a nivel de palabra, por lo que direccionará 64M palabras de 16 bits, y por tanto el bus de direcciones será de 26 bits. Además tiene 8 registros internos, por lo que se necesitan 3 bits para codificar cada registro.

Con esto se puede plantear la siguiente codificación de las instrucciones planteadas:

<table>
<tr><td></td><td>31</td><td></td><td></td><td></td><td></td><td></td><td></td><td>24</td><td>23</td><td></td><td></td><td></td><td></td><td></td><td></td><td>16</td><td>15</td><td></td><td></td><td></td><td></td><td></td><td></td><td>8</td><td>7</td><td></td><td></td><td></td><td></td><td>0</td></tr>
<tr><td>G. I</td><td>0</td><td>0</td><td>op</td><td colspan="3">Rd</td><td colspan="24">Direccionamiento directo del dato</td></tr>
<tr><td>G. II</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>1</td><td>1</td><td>1</td><td>0</td><td colspan="3">opcode</td><td colspan="3">Rd</td><td colspan="3">Rb</td><td>Ri</td></tr>
<tr><td>G. III</td><td>0</td><td>1</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>op</td><td colspan="3">Rd</td><td colspan="3">Rb</td><td colspan="8">Desplazamiento 10 bits</td></tr>
<tr><td>G. IV</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>1</td><td>1</td><td>1</td><td>1</td><td>0</td><td>X</td><td colspan="4">opcode</td><td colspan="3">Rd</td><td>Ra</td></tr>
<tr><td>G. V</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>1</td><td>1</td><td>0</td><td colspan="2">Op</td><td colspan="3">Rd</td><td colspan="6">dato</td></tr>
<tr><td>G. VI</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>1</td><td>1</td><td>1</td><td>1</td><td>1</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td>X</td><td colspan="2">opcode</td></tr>
<tr><td>G. VII</td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td></td><td>1</td><td>0</td><td colspan="3">opcode</td><td colspan="9">Desplazamiento dirección +/- 1K</td></tr>
</table>

2)

a) Número máximo de palabras de una instrucción: 2 palabras, tal y como se ha podido ver con las instrucciones del grupo I y III.

b) Número mínimo de palabras de una instrucción: 1 palabra, tal y como se ha podido ver en las instrucciones de los grupos II, IV, V, VI y VII.

c) IR de 32 bits, ya que debe ser capaz de albergar toda una instrucción completa. Por lo tanto tiene que tener el tamaño de la instrucción más grande.

d) PC de 26 bits. Es una dirección de memoria de programa, por lo que será del mismo tamaño del bus de direcciones de dicha memoria.

e) SP de 26 bits. Es una dirección de memoria de datos, por lo que será del mismo tamaño del bus de direcciones de dicha memoria.

f) Registros internos de 26 bits (máximo entre direcciones de datos y datos)

Ejercicio 3 (30 minutos – 3 puntos)

Se quiere desarrollar una aplicación con el microcontrolador STM32L152RB para una cafetera que incluya un **LED, que se enciende cuando se le aplica un '1' lógico**, y un selector de modo de funcionamiento, tal como se muestra en la figura 3. Para la conexión de estos dos dispositivos se van a utilizar los pines **PC0, PC1, PC2 y PC3** del microcontrolador.

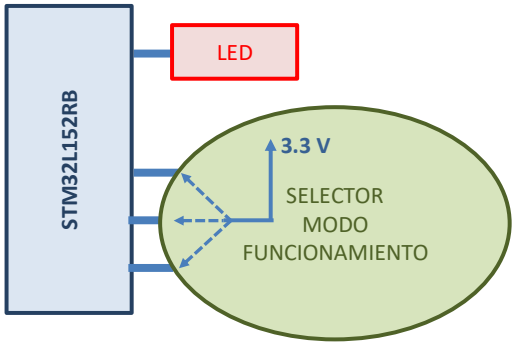


Figura 3

El selector de modo de funcionamiento puede estar en una de estas tres posiciones:

- APAGADO: En esta posición el LED está apagado
- CAFÉ EXPRESSO: En esta posición el LED está encendido
- CAFÉ CAPPUCHINO: En esta posición el LED está parpadeando

Al final de este enunciado se encuentra el código desarrollado para la aplicación.

Se pide:

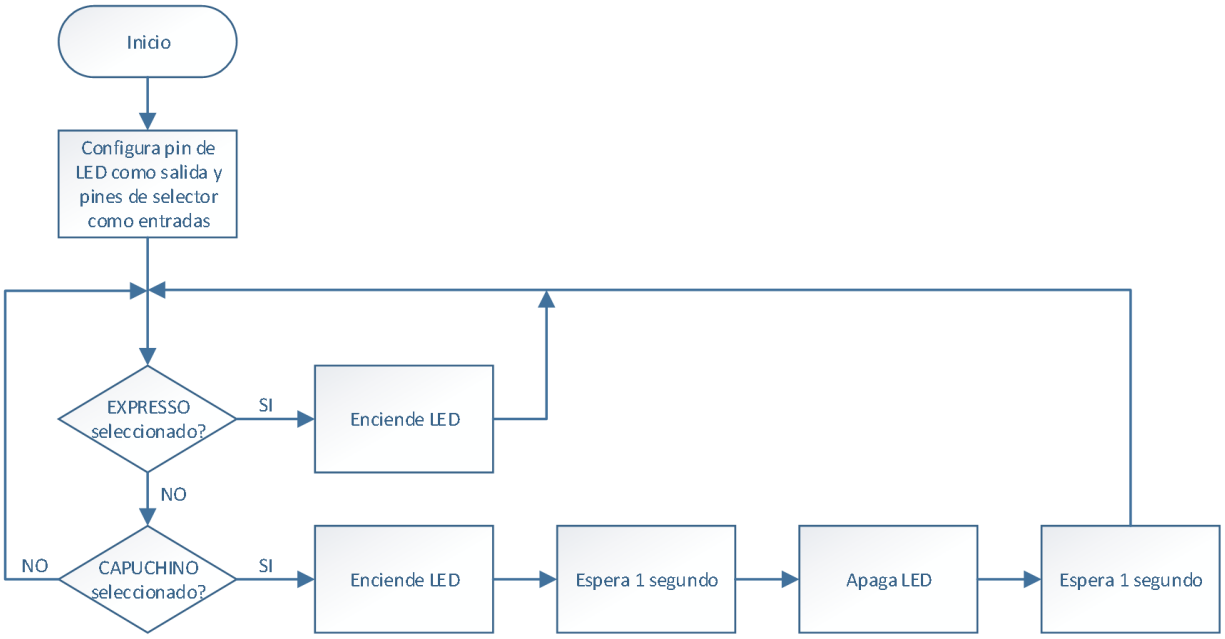
a) Suponiendo que antes de iniciar la ejecución del programa se resetea el microcontrolador, indique el contenido de los registros que se indican después de la ejecución de las líneas de inicialización del programa (líneas de código anteriores a while(1)) *(2 puntos)*

REGISTRO	CONTENIDO EN HEXADECIMAL
GPIOC_MODER	*En binario: xxxx xxxx xxxx xxxx xxxx xxxx 0100 0000* *Considerando, tal cual pone en el datasheet, los valores iniciales del registro tras el reset, que es con todos los bits a '0'.* *En hexadecimal: 0x00000040*

b) Indique razonadamente a qué pines se encuentran conectados el LED y el selector de modo de funcionamiento *(2 puntos)*

Dispositivo		**Pines de conexión**	**Justificación**
LED		*PC3*	*Es el único pin que se ha configurado como salida digital*
Selector de modo de funcionamiento	APAGADO	*PC0*	*Por descarte, es el último pin configurado como entrada y con sólo funcionalidad de apagar el led cuando no están activos los otros dos*
	CAFÉ EXPRESSO	*PC1*	*Pin configurado como entrada, y que si se activa, el led se queda encendido*
	CAFÉ CAPPUCHINO	*PC2*	*Pin configurado como entrada y que si se activa parpadea el led*

c) Represente el diagrama de flujo del programa *(3 puntos)*



d) Indique qué cambios realizaría en el código si se cambia el pin de conexión del LED al pin PB6 (ponga una cruz al lado de las líneas de código que habría que cambiar e incluya a la derecha la nueva línea para este cambio en la conexión del LED). *(3 puntos)*

En lugar de poner el PC3 como salida, habría que poner el PB6. Además, en lugar de apagar y encender el PC3, habrá que hacer lo mismo pero con el PB6. Código nuevo en azul.

CÓDIGO DEL PROGRAMA

```
#include "stm32l1xx.h"
# include "retardo.h" // BIBLIOTECA QUE INCLUYE LA FUNCIÓN RETARDO (X)
                      // RETARDO(X) PRODUCE RETARDO DEL Nº DE SEGUNDOS INDICADO EN X


int main(void){

// INICIALIZACIÓN GPIO
GPIOC->MODER &= ~0x0000003F;
GPIOC->MODER &= ~(1 << (3*2 +1));          *   GPIOB->MODER &= ~(1 << (6*2 +1));
GPIOC->MODER |=(1 << (3*2));               *   GPIOB->MODER |=(1 << (6*2));

while (1) {
        if ((GPIOC->IDR&0x00000002)!=0) {
                GPIOC->BSRR = (1<<3);      *   GPIOB->BSRR = (1<<6);
        }
        else if ((GPIOC->IDR&0x00000004)!=0) {
                GPIOC->BSRR = (1<<3);      *   GPIOB->BSRR = (1<<6);
                retardo(1);
                GPIOC->BSRR = (1<<3)<<16;  *   GPIOB->BSRR = (1<<6)<<16;
                retardo (1);
        }
        else {
                GPIOC->BSRR = (1<<3)<<16;  *   GPIOB->BSRR = (1<<6)<<16;
        }
}
}
```