EJERCICIO 1 (2.0 puntos):

En una arquitectura Harvard, con una memoria de programa de 16Kx16 y una memoria de datos de 64KB siendo los datos de 8 bits, y considerando que la CPU tiene 6 registros internos para datos, y que se implementa siguiendo una filosofía Load & Store:

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. Los tipos de instrucciones que debe tener el microprocesador, son:

- 3 instrucciones de transferencia de datos entre memoria y registros internos, con direccionamiento absoluto.
- 8 instrucciones aritmético/lógicas de operar entre registros, indicando en la instrucción tanto los registros operandos, como el registro que almacena el resultado.
- 4 instrucciones aritmético/lógicas con un operando dado con direccionamiento inmediato, y donde el otro operando y el registro de resultado son el mismo.
- 3 instrucciones de control con direccionamiento inherente
- 5 saltos condicionales con direccionamiento relativo a contador de programa, siendo el desplazamiento relativo de más/menos 512B

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

a) Número mínimo de palabras que usa una instrucción

b) Número máximo de palabras que usa una instrucción

c) Tamaño del Registro de Instrucción

d) Tamaño del Contador de Programa

e) Tamaño de los Registros internos

EJERCICIO 2 (4.0 puntos):

Se ha desarrollado un programa para un dispositivo con un microcontrolador SMT32L5152RB (el programa figura en el Anexo I). Se sabe que se han conectado los siguientes periféricos al dispositivo:

PB0 Led blanco	PA0 Pulsador blanco
PB1 Led amarillo	PA1 Pulsador amarillo
PB2 Led azul	PA2 Pulsador azul
PB3 Led naranja	PA3 Pulsador naranja
PB4 Led rojo	PA4 Pulsador negro
PB5 Led verde	

Además se ha creado una función (invocada desde el programa) que se define como:

```
unsigned char Aleatorio(void);
```

Cuando esta función es invocada devuelve un número pseudoaleatorio comprendido entre 0 y 3.

Analiza el programa y responde a las siguientes cuestiones, indicando qué hace cada fragmento de código de una forma funcional, no describiendo cada instrucción (decir que var1 ++ incrementa var1 es algo trivial, hay que describir qué funcionalidad tiene var1 y para qué se incrementa), si una variable sólo puede tomar unos valores indica que valores son. Si algunas acciones tienen una duración definida, indícalo:

a) ¿Qué hace TIM4_IRQHandler? ¿Cuándo es llamada? (1/10).
b) Describe como están inicializados los periféricos (funcionalidad) (1/10).
c) ¿Qué se hace en el fragmento de código con el comentario `//(1)` hasta el `//(2)` no incluido? (1/10).
d) ¿Qué se hace en el fragmento de código con el comentario `//(2)` hasta el `//(3)` no incluido? (1/10).
e) ¿Qué se hace en el fragmento de código con el comentario `//(3)` hasta el final? (1/10).
f) ¿Qué hacen Func1 y Func2? (1/10).
g) Describe de una forma general qué hacen el programa y el dispositivo (3/10).

EJERCICIO 3 (4.0 puntos):

Con un microcontrolador STM32L152RB con pclk a 12MHz, sin tener por qué utilizar la placa STM32L_Discovery, se quiere diseñar un sistema de control de iluminación para una sala de fiestas. Los requisitos que nos pone el cliente son los siguientes:

- El control de iluminación estará compuesto por 4 juegos de luces.
- Cada juego de luces se controla con una misma señal PWM de 100KHz, cuyo duty cycle debe estar siempre entre el 10% y el 85%.
- La potencia luminosa de cada juego de luces responderá linealmente la intensidad sonora procedente de una banda de frecuencias:
  - Juego J1: 50 – 800 Hz
  - Juego J2: 800 – 4000 Hz
  - Juego J3: 4000 – 10000 Hz
  - Juego J4: 10000 – 20000 Hz
- Se dispone de unos filtros paso-banda externos que proporcionan a su salida un valor de continua de la intensidad sonora cada banda, siendo dicho valor de continua entre 0 y 3V.
- Adicionalmente se necesita tener un pulsador por cada juego de luces que, durante el tiempo que esté pulsado, ponga el juego de luces pulsado a máxima potencia, y retorne a su valor anterior una vez soltado el pulsador.

Con esta información, y aquellas aproximaciones o decisiones que Vd. considere necesario (siempre justificándolas) conteste a las siguientes preguntas:

a) Realice el diagrama de bloques de la solución
b) Indique si utilizaría interrupciones, y en caso afirmativo, para qué y con qué funcionalidad.
c) Configure los periféricos utilizados
d) Realice el diagrama de flujo de la solución
e) Si además se quisiera que las luces lucieran siempre de forma intermitente, con intervalos de tiempos modificables, ¿Cómo modificaría el diseño realizado?

Anexo I

```
#include "../stm32l1xx.h"
#include "../Biblioteca_SDM.h"
#include "../Utiles_SDM.h"
#include<math.h>

unsigned char tiempo = 0;

void Func1(unsigned char* array) {
    int i;
    for (i = 0; i < 20; i++)
        array[i] = pow(2,Aleatorio()); // La función pow(x,y) calcula la potencia x^y
}
void Func2(unsigned char* array, unsigned char mostrados) {
    int i;
    for (i = 0; i <= mostrados; i++) {
        tiempo = 20;
        GPIOB->ODR &= array[i];
        while (tiempo);
        tiempo = 2;
        GPIOB->ODR &= 0;
        while (tiempo);
    }
}
void TIM4_IRQHandler(void) {
    if ((TIM4->SR & 0x0002)!=0) {
        TIM4->SR &= ~0x0002;
        TIM4->CNT = 0;
        TIM4->CCR1 = 100;
        if (tiempo)
            tiempo --;
    }
}
int main(void){
    unsigned char Lista[20];
    unsigned char mostrados, pulsaciones;
    int valor, i;

    GPIOA->MODER = (0x00000000);
    GPIOB->MODER = (0x00000555);
    // El reloj que llega a TIM4 es de 32 MHz
    TIM4->CR1 = 0x0000;
    TIM4->CR2 = 0x0000;
    TIM4->SMCR = 0x0000;
    TIM4->PSC = 32000;
    TIM4->CNT = 0;
    TIM4->ARR = 0xFFFF;
    TIM4->CCR1 = 100;
    TIM4->DCR = 0;
    TIM4->DIER = 0x0002;
    TIM4->CCMR1 = 0x0000;
    TIM4->CCER = 0x0000;
    TIM4->CR1 |= 0x0001;
    TIM4->EGR |= 0x0001;
    TIM4->SR = 0;
    NVIC->ISER[0] |= (1 << 30);
```

```
/*****************************************************************************
    Comienzo del bucle principal
*****************************************************************************/
    while (1) {
        Func1(&Lista[0]);
        while (GPIOA->IDR & 0x0010);                                    // (1)
        while ((GPIOA->IDR & 0x0010) == 0);
        mostrados = 0;
        while (mostrados < 20) {                                        // (2)
            Func2(&Lista[0], mostrados);
            pulsaciones = 0;
            while (pulsaciones <= mostrados) {
                while ((GPIOA->IDR & 0x000F) == 0);
                if ((GPIOA->IDR & 0x000F) == Lista[mostrados])
                    pulsaciones ++;
                else {
                    mostrados = 19;
                    pulsaciones = 30;
                }
                while (GPIOA->IDR & 0x000F);
            }
            mostrados ++;
        }
        if (pulsaciones == 30)                                          // (3)
            valor = 0x0010;
        else
            valor = 0x0020;
        for (i=0; i<=4; i++) {
            tiempo = 10;
            GPIOB->ODR &= valor;
            while (tiempo);
            tiempo = 10;
            GPIOB->ODR = 0;
            while(tiempo);
        }
    }
}
```