

CUESTIONES:

C1. Enumera y comenta brevemente las etapas de que consta el proceso de generación de programas para un sistema empotrado (proceso de producción del software). Indica las herramientas utilizadas. (0.5 puntos).

Solución:

El proceso de generación de programas aparece en la página 2 del tema 2. El primer proceso una vez realizada la edición con el editor de texto del programa es el compilador de lenguaje C o bien el ensamblador dependiendo del lenguaje usado en el desarrollo. Una vez obtenido el fichero objeto resultado de la compilación, el linkador se encarga de enlazar todos los módulos que se hayan empleado para realizar el programa.

Una vez disponible el programa reubicable resultado del enlazado de módulos, el programa locator se encarga de convertirlo en un programa absoluto que será colocado en direcciones de memoria absolutas del sistema de desarrollo.

C2. Explica cómo se emplean las librerías en un programa y cuál es el contenido de los ficheros cabecera con la extensión 'h' que acompañan a las librerías. (0.5 puntos).

Solución:

Las librerías se deben enlazar conjuntamente con el programa para poder ser usadas. De esto se encarga la gestión de proyectos del entorno de desarrollo que estemos usando. Se puede incluir el modulo en código fuente de la librería o bien el código compilado en formato obj o lib. Además de esto se debe incluir el "fichero cabecera" en el código de nuestro programa usando la directiva del preprocesador "#include".

El contenido de los ficheros cabecera son los prototipos de las funciones que puede usar el usuario, junto a la definición de constantes que emplee la librería.

C3. Indica las diferencias entre las sondas de acceso al diseño y los emuladores en circuito empleados para la depuración de sistemas empotrados. (0.5 puntos).

Solución:

Tanto las sondas de acceso al diseño como los emuladores en circuito son soluciones hardware para la depuración de programas en el sistema final del usuario. El emulador en circuito es una solución más potente pero más cara. Con ella es posible visualizar todos los registros internos y el estado del microcontrolador y suelen tener muchas más funciones.

La sonda de acceso al diseño es una solución más limitada para la depuración del hardware, pero permite como mínimo revisar los registros internos del procesador, su estado y ejecutar paso a paso el programa. Todo esto a costa de usar algunos puertos de E/S del microcontrolador para la conexión a la sonda, que ya no se podrán usar para otras funciones.

C4. Indica cómo harías para extraer los bits 0,1 y 2 de la variable valor y copiarlos a otra variable tmp de tal forma que el resultado estuviese acotado entre 0 y 7. (0.5 puntos).

Solución:

```
tmp=valor&0x07;
```

C5. Explica como funcionan las instrucciones de salto condicional del PIC **BTFS** y **BTFS**. (0.5 puntos)

Solución:

Las instrucciones de salto condicional tienen la sintaxis:

BTFS(S/C) puerto,bit

Saltan la siguiente instrucción si el bit del puerto esta a uno o a cero respectivamente.

PROBLEMAS:

P1. (2.5 puntos) El siguiente código fuente C para PIC a 4 MHz pretende generar una señal de 1KHz con el Timer 0 sin usar el preescalador. Tiene varios errores, tanto de sintaxis como semánticos. Encontrarlos y corregir el código fuente.

```
#include <pic.h>

unsigned char fase;
void interrupt irs(void) {
if (T0IF) {
    if (!(fase%4)) RC0=!RC0;
    fase++;
    TMR0=(256-100);
}
}
void main(void) {
fase=0;
PSA=1; PS2=0; PS1=0; PS0=1;
TMR0=(256-125)
T0CS=0; T0IE=1; GIE=1;
while(1) {
}
}
```

Solución:

```
#include <pic.h>
unsigned char fase;
void interrupt irs(void) {
if (T0IF) {
    if (!(fase%4)) RC0=!RC0;
    fase++;
    TMR0=(256-125); //125*4=500useg-> 1KHz
}
    T0IF=0; //bajar la bandera
}
}
```

```

void main(void) {
    TRISC0=0; //configurar salida
    fase=0;
    PSA=1; PS2=0; PS1=0; PS0=1; //no es necesario
    TMR0=(256-125); //falta punto y coma
    T0CS=0; T0IE=1; GIE=1;
    while(1) {
        }
    }
}

```

P2. (3 puntos) La empresa TIRSA S.A. ha encargado el desarrollo de una pequeña maquina tragaperras según las siguientes especificaciones:

1. La palanca de inicio de juego estará en el puerto RB1, El pulsador de moneda en el puerto RB0 ambos activos a nivel lógico alto. Cada vez que se detecta moneda se guarda como crédito disponible y se queda a la espera de pulsar la palanca de inicio.
2. El intervalo de tiempo que transcurre entre la pulsación de la palanca y su vuelta a la posición inicial puede emplearse para generar tres números aleatorios comprendidos entre 0 y 255. Esto lo hace la función void genera_numeros(unsigned char *n1, unsigned char *n2, unsigned char *n3); Como tiene que devolver tres variables, se usan los parámetros pasados por referencia con punteros.
3. Una luz intermitente conectada a RB2 se emplea para indicar que la máquina se está “moviendo” mientras la palanca está accionada.
4. Hay cuatro símbolos de frutas que serán mostradas en cada una de las tres columnas de la pantalla de la máquina.

```

#define FRESA 0
#define PLATANO 1
#define PERA 2
#define LIMON 3

```

5. Existe una función que no hay que desarrollar para mostrar esta información en pantalla: void display(unsigned char column, unsigned char símbolo). Todos los índices empiezan en cero.
6. El pago de la máquina se calcula según las siguientes reglas:
 - a. Tres figuras iguales de cualquier tipo obtiene 10 monedas
 - b. Una fresa en cualquier lugar obtiene 1 moneda.
 - c. Tres fresas obtienen 20 monedas
 - d. Cualquier otra combinación pierde.
7. Existe una función que no hay que desarrollar para pagar: void pagar(unsigned char num_monedas).

Se pide:

- a) Escribir el diagrama de flujo y el programa principal del sistema. No es necesario usar interrupciones.
- b) Escribir la función para obtener el número aleatorio “genera_numeros” teniendo en cuenta el tiempo que se ha pulsado el pulsador de inicio. Para calcular los tres números aleatorios se van actualizando, mientras este bajada la palanca, tres contadores mediante la operación or-exclusiva realizada con tres constantes diferentes para cada columna, por ejemplo para una variable aleatoria: $var1 = var1 \oplus 0x34$;

Solución:

a) El diagrama de flujo es muy simple. Primero se espera que entren monedas y se van anotando las monedas entrantes. Cuando se baja la palanca y hay monedas se realiza la generación de números aleatorios mientras la palanca esta bajada, cuando se suelta se termina de generar los números aleatorios y se representa el resultado. Si hay premio se entrega al cliente y se vuelve a esperar a que se baje la palanca o se introduzcan más monedas.

```
void main(void) {
unsigned char n1,n2,n3,monedas;

monedas=0;
while (1) {
    while (!RB1 || !monedas) //mientras no se pulse y no haya monedas
        if (RB0) {
            monedas++;
            while (!RB0); //espera que entre moneda
        }
    //se ha pulsado la palanca
    //la funcion genera numeros espera que se suelte y devuelve los numeros
    genera_numeros(&n1,&n2,&n3);
    monedas--;
    display(0,n1>>6); //saca por pantalla resultado
    display(1,n2>>6);
    display(2,n3>>6);
    //los premios se acumulan
    if ((n1==n2) && (n2==n3) && (n1==n3)) pagar(10); //tres iguales
    if ((n1==FRESA) || (n2==FRESA) || (n3==FRESA)) pagar(1); //una fresa
    if ((n1==FRESA) && (n2==FRESA) && (n3==FRESA)) pagar(20); //tres fresas
}
}
```

b) Los números se calculan continuamente acumulando los contadores hasta que se sube la palanca...

```
void genera_numeros(unsigned char *n1,unsigned char *n2, unsigned char *n3)
{
while (RB1) { //mientras este pulsada la palanca se van acumulando los
numeros aleatorios
    *n1+=*n1^0x12;
    *n2+=*n2^0x34;
    *n3+=*n3^0x56;
}
}
```

P3. (2 puntos) Realizar la función “void imprime_char (unsigned char valor)” que emplee la función de escritura de un carácter ASCII en la pantalla LCD: lcd_putchar(). La función debe imprimir una variable de 8 bits en formato binario, o sea debe sacar 0b seguido de ocho caracteres que pueden ser ‘0’ o ‘1’ en función del número que se le pase a la función. Por ejemplo, si la variable valor contiene el numero 45 decimal, mostraría por pantalla: “0b00101101”.

Solución:

```
void imprime_char (unsigned char valor) {
    unsigned char i;

    for (i=7;i>=0;i--)
        if (valor & (1<<i)) lcd_putchar('1');
        else lcd_putchar('0');
}
```