

**Nota: El examen resuelto puede tener algún error que se haya deslizado.**

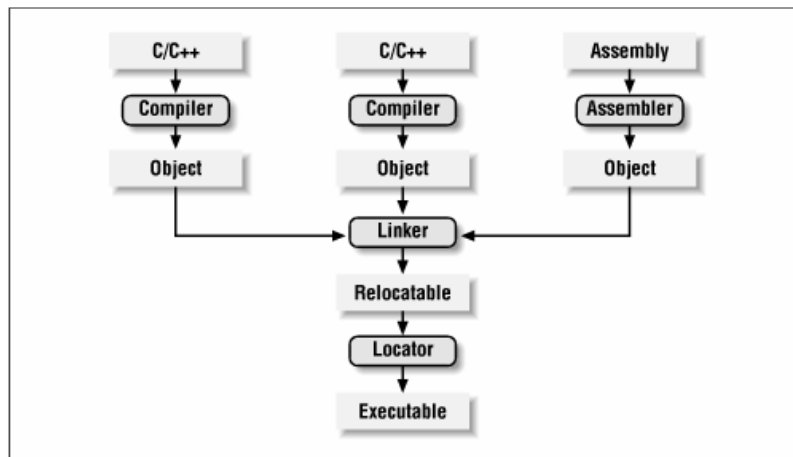
**Por favor no tomarlo al pie de la letra y preguntar en tutoría las dudas que se tengan sobre la resolución.**

## CUESTIONES:

C1. Enumera y comenta brevemente los procesos que tienen lugar y las herramientas empleadas durante la creación de programas para un sistema empujado (0.5 puntos).

Solución:

*El proceso de convertir la representación en código fuente del software empujado en una imagen binaria ejecutable involucra tres pasos distintos. Primero, cada uno de los ficheros fuente deben ser compilados o ensamblados en un fichero objeto. Segundo, todos los ficheros objeto que resultan del primer paso deben linkarse juntos para producir un fichero objeto único, denominado **programa reubicable**. Finalmente, direcciones de memoria física se asignan a los desplazamientos relativos del programa reubicable en un proceso denominado reubicación. El resultado de este tercer paso es un fichero que contiene una imagen binaria ejecutable que está lista para ejecutarse en el sistema empujado.*



C2. Indica como se indica al compilador HI-TECH del PIC la colocación de variables y constantes en memoria de código y de datos. (0.5 puntos).

Solución:

*El compilador coloca en memoria de código cualquier variable a la que se le anteponga el **modificador "const"**. Lógicamente esta variable no puede ser modificada en el transcurso del programa. El resto de variables se colocan siempre por defecto en memoria de datos.*

C3. Indica las diferencias entre los cargadores y los monitores de sistema empleados para el desarrollo de programas en sistemas empotrados. (0.5 puntos).

Solución:

Los monitores de sistema o residentes son soluciones software frente a los emuladores en circuito que son soluciones hardware. Cuando no se tiene acceso físico al sistema y/o las tareas de depuración son simples se suelen emplear monitores residentes que son soluciones más económicas pero que tienen limitaciones, al no permitir el acceso completo al estado del microcontrolador. También interfieren con el programa a testear porque utilizan memoria y recursos hardware como patillas del microcontrolador que ya no se podrán usar para otras tareas.

El cargador consiste en dos programas, uno se ejecuta en el sistema microcontrolador y el otro en el ordenador de desarrollo. El programa del sistema microcontrolador permite la carga del programa que queramos probar en la memoria de código del sistema de desarrollo a través de un enlace de comunicación de algún tipo (USB, SERIE) con el ordenador de desarrollo en el que se ejecuta el otro programa.

El cargador solo se limita a la carga y ejecución del programa y no permite comandos de depuración como la ejecución paso a paso y demás comandos de depuración que si permite el monitor residente, aunque a costa de emplear recursos de la CPU. El cargador se usa solo para cargar el programa y también ocupa menos recursos de la CPU.

C4. Indica cómo harías para extraer los bits 4 y 5 de la variable valor y copiarlos a otra variable tmp de tal forma que el resultado estuviese acotado entre 0 y 3. (0.5 puntos).

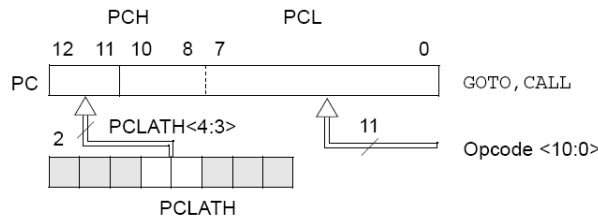
Solución:

```
tmp=((var>>4)&0x03);
```

C5. Explica como funciona y como se emplea las instrucción en ensamblador de salto incondicional del PIC **GOTO**. (0.5 puntos)

Solución:

La instrucción **GOTO** del PIC permite realizar saltos incondicionales a cualquier parte del código. Para realizar el salto la instrucción solo almacena 11 bits de la dirección destino. Los restantes dos bits necesarios para direccionar toda la memoria de código (bits 11 y 12) se extraen del registro PCH que almacena los bits altos de la dirección de memoria actual.



```
: bucle
-----
-----
GOTO bucle
```

## PROBLEMAS:

P1. (2.5 puntos) El siguiente código fuente C para PIC a 4 MHz pretende sacar por pantalla un tipo de datos “unsigned char” en formato decimal. Tiene varios errores, tanto de sintaxis como semánticos. Encontrarlos y corregir el código fuente.

### Solución:

```
/* saca el valor del char en decimal quitando los ceros
   menos significativos */

void putchdec(unsigned char c)
{
    unsigned char temp;

    temp=c;
    /* unidades de centena */
    if ((c/100)>0) putch((c/100)+'0');
    else putch(' ');
    c--=(c/100)*100;
    /* decenas */
    if (((temp/10)>0) || ((temp/100)>0)) putch((c/10)+'0');
    else putch(' ');
    c--=(c/10)*10;
    /* unidades */
    putch(c+'0');
}
}
```

P2. (3 puntos) La empresa CALOR Y FRIO S.A. ha encargado el desarrollo de sistema de medida de temperatura de cuatro cámaras “secaderos” y alarma con cuatro sensores basado en un PIC 16F876 según las siguientes especificaciones:

1. El aparato dispone de cuatro sensores conectados a las patillas RA0 a RA3.
2. Otra empresa se ha encargado del desarrollo del software de medida en forma de librería ADC. Previa inicialización con la función void adc\_init(void), la medida se realiza a través de la función **adc\_read**, a la que se le pasa el canal a convertir (0-3) y devuelve un entero con la lectura realizada (0-1023): unsigned int adc\_read(unsigned char).
3. El sensor devuelve 10mV por grado centígrado, por lo que a 25°C, la lectura es de 250mV. El convertidor devuelve el código 0 para 0 voltios y el código 1024 para 5Voltios.
4. La alarma se dispara cuando cualquiera de las cuatro medidas alcanza el valor de 60 grados centígrados y sonará durante 60 segundos (**a temporizar con el timer**) si no se desconecta antes. La alarma está conectada a la **patilla RA4** y se activa con un nivel lógico ‘1’. Se puede desconectar pulsando la tecla ‘STOP’ conectada a la **patilla RA5**.



---

```

resultado=((unsigned long)tension*500)/1024;
if (resultado>99) resultado=99; //acoto temperatura

return resultado;
}

void main(void) {

unsigned char sec0,sec1,sec2,sec3;
//temperaturas de cada secadero en grados centigrados

TRISA=0xEF; // 1110 1111 RA4 salida, RA5 entrada, demas
entradas
TRISB=0x00; //todo salidas
TRISC=0x00; //todo salidas

RA4=0;
adc_init();

T0CS=0; //selecciono reloj interno
PSA=0; //activo preescaler por 256
PS0=1; PS1=1; PS2=1;
// 256*256*916 aprox= 60 000 000 microseg.
T0IE=0; //Desactivo la interrupción del timer
GIE=1; //Activo todas las interrupciones

while(1) {
    sec0==c_cent(adc_read(0)); //leo y convierto secadero 0
    sec1==c_cent(adc_read(1)); //leo y convierto secadero 1
    sec2==c_cent(adc_read(2)); //leo y convierto secadero 2
    sec3==c_cent(adc_read(3)); //leo y convierto secadero 3
    // refresca y muestra temperaturas
    refresca(sec0,sec1,sec2,sec3);
    // comprueba si sube la temperatura
    if ((sec0>=60) || (sec1>=60) || (sec2>=60) || (sec3>=60))
    {
        RA4=1;
        contador=916; //inicializa el contador
        T0IE=1; //activo interrupción
    }
    // si pulsa boton STOP se para alarma y la interrupcion
    if (!RA5) {
        RA4=0;
        T0IE=0; //desactivo interrupción
    }
}
}

```

---

b) Escribir la función o funciones para leer el teclado y refrescar el display.

Solución:

---

```
// tabla de control de 7 segmentos. el segmento a está en
RB0, el b en RB1 y así sucesivamente
// para encender el segmento la tabla pone un 1 lo que encaja
con el tipo de cátodo común
// y no es necesaria más modificación
unsigned char tabla_7seg[]={0x3F,0x06,
0x5B,0x4F,0x66,0x6D,0x7D,0x07,0x7F,0x6F,0x77,0x7C,0x39,0x5E,0x
79,0x71};

void delay(unsigned char valor) {
unsigned int tiempo;
// temporizo valor miliseg. con el timer 1
// se puede hacer también de forma aproximada con bucles for

    TMR1CS=0; TMR1ON=1; T1CKPS0=0; T1CKPS1=0;
    TMR1IF=0; //baja la bandera
    tiempo=((unsigned int)valor)*1000;
    //temporiza hasta 65 milisegundos
    TMR1H=(65536-tiempo)>>8; //parte alta del entero
    TMR1L=(65536-tiempo)&0x00FF; //parte baja del entero
    while (!TMR1IF); //espera a que pase el tiempo

}

void refresca(unsigned char sec0, unsigned char sec1, unsigned
char sec2, unsigned char sec3) {
unsigned char tmp;

PORTB=0x00; //Apago todos los displays
//comienzo por las unidades del secadero 0 más a la izq
PORTB=tabla_7seg[sec0%10]; //pongo unidades
RC0=1; //activo transistor
delay(6); //espero
RC0=0;
//sigo con las decenas
PORTB=tabla_7seg[sec0/10]; //pongo decenas
RC1=1; //activo unidades
delay(6); //espero
RC1=0;
//sigo con las unidades del secadero 1 siguiente a la izq
PORTB=tabla_7seg[sec1%10]; //pongo unidades
RC2=1; //activo transistor
delay(6); //espero
RC2=0;
//sigo con las decenas
PORTB=tabla_7seg[sec1/10]; //pongo decenas
RC3=1; //activo unidades
delay(6); //espero
```

---

---

```

RC3=0;
//sigo con las unidades del secadero 2 siguiente a la izq
PORTB=tabla_7seg[sec2%10]; //pongo unidades
RC4=1; //activo transistor
delay(6); //espero
RC4=0;
//sigo con las decenas
PORTB=tabla_7seg[sec2/10]; //pongo decenas
RC5=1; //activo unidades
delay(6); //espero
RC5=0;
//sigo con las unidades del secadero 3 ultimo a la derecha
PORTB=tabla_7seg[sec3%10]; //pongo unidades
RC6=1; //activo transistor
delay(6); //espero
RC6=0;
//sigo con las decenas
PORTB=tabla_7seg[sec3/10]; //pongo decenas
RC7=1; //activo unidades
delay(6); //espero
RC7=0;
}

```

---

P3. (2 puntos) Realizar la función “void imprime\_char (signed 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 decimal **CON SIGNO**, o sea debe sacar un máximo de tres caracteres y signo que pueden ser desde el ‘0’ hasta el ‘9’ en función del número que se le pase a la función. Por ejemplo, si la variable valor contiene el numero 45 en decimal, mostraría por pantalla: “045” si tuviese el número -10, mostraría: “-010”.

Solución:

---

```

void imprime_char(signed char valor) {
// si es negativo pongo positivo
if (valor<0) {
    lcd_putchar('-');
    valor=valor*(-1);
}
//saco centena
lcd_putchar ((valor/100) +'0');
//saco decena
lcd_putchar ((valor/10)%10 +'0');
//saco unidad
lcd_putchar ((valor%10) +'0');
}

```

---