UNIVERSIDAD CARLOS III DE MADRID (Dpto. de Tecnología Electrónica)
Sist. Dig. Basados en Microprocesador **(Gr. Ing. Telemática)**
6 de mayo de 2014 2º PARCIAL (100 minutos)

PROBLEMA 1 (2.5 puntos): *SOLUCIÓN*

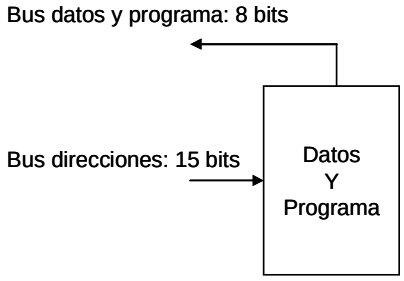
Diseñe un microprocesador con las siguientes características:

- 16KB de memoria de datos.
- 8KB de memoria de programa.
- Juego de 128 instrucciones de 16 bits.
- Bus de datos de 8 bits.
- Minimice la velocidad de carga de la instrucción (Fetch).

En caso de implementarse con arquitectura Von Neuman, se pide:

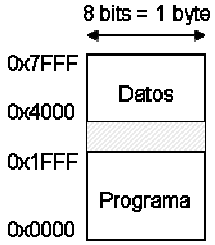
a. Diagrama de bloques indicando el tamaño de los buses principales. (0,75 puntos)

En esta arquitectura la memoria de programa y de datos están juntas, como el bus de datos es de 8 bits, esto obliga a que el bus de programa también lo sea y por tanto el tamaño de palabra será de 1 byte.
Como necesitamos 16KB de datos y 8KB de programa, tenemos una necesidad de 24KB. No podemos direccionar 24KB, tendremos que irnos a la primera potencia de 2 que esté por encima de 24KB, es decir, 32KB.
Por tanto:
- Bus de direcciones (32KB): 15 líneas de dirección (bits).
- Bus de datos/programa: 8 bits



b. Mapa de memoria (1 punto).

32KB de memoria en total -> 15 líneas -> 0x7FFF
16KB de datos -> 14 líneas -> 0x3FFF
8KB de programa -> 13 líneas -> 0x1FFF
Juego de 128 instrucciones de 16 bits -> No es un condicionante de diseño para la memoria, sino una característica interna. Estas instrucciones, sean como sean, se almacenarán en la memoria de programa.



c. Tamaño del contador de programa y del registro de instrucción (0,75 puntos).

PC: 15 bits (los necesarios para direccionar 32KB), si bien también podría ser de 13 bits limitando el acceso sólo a la memoria ROM (es decir, ni se podría ampliar la memoria de programa del dispositivo, ni se podría utilizar la memoria de datos para aplicaciones).
RI: 16 bits (tamaño de la instrucción).

UNIVERSIDAD CARLOS III DE MADRID (Dpto. de Tecnología Electrónica)
Sist. Dig. Basados en Microprocesador **(Gr. Ing. Telemática)**
6 de mayo de 2014 2º PARCIAL (100 minutos)

PROBLEMA 2 (2.5 puntos): *SOLUCIÓN*

Se tiene un microprocesador con las siguientes características:

- Arquitectura Harvard
- 16 registros internos que pueden ser utilizados tanto para direcciones como para datos, sin poder acceder a la memoria de programa.
- 1KB de datos organizados en palabras de 8 bits
- Capacidad de direccionar 8K instrucciones
- Filosofía Load & Store

1. Diseñe una codificación de los distintos tipos de instrucciones microprocesador, minimizando en lo posible el tamaño de la palabra de instrucción. Los tipos de instrucciones que debe tener el microprocesador, son (1,5 puntos):

- 9 instrucciones aritmético/lógicas de operar entre registros, contemplando que el resultado siempre se guarda en uno de los dos registros de los operandos
  - Ejemplo: ADD Rd, Rs; Rd ← Rd + Rs
- 6 instrucciones aritmético/lógicas con direccionamiento inmediato, donde el resultado se almacenará en el registro del otro operando.
  - Ejemplo: ADDI Rd, 115; Rd ← Rd + 115
- 8 instrucciones de salto absoluto condicional
  - Ejemplo: JMP <direccion>; PC ← <direccion>
- 7 instrucciones de transferencia de información, con direccionamiento directo
  - Ejemplo: LD Rd, <direccion>; Rd ← (<direccion>)
- 7 instrucciones de transferencia de información, con direccionamiento indirecto
  - Ejemplo: LDI Rd, Ri; Rd ← (Ri)

2. Describa con el mayor detalle posible la ejecución de la siguiente instrucción en RTL, si se ejecutase en un microcontrolador con arquitectura de 32 bits y memoria de programa de 32 bits (1 punto).

ADD Rd, Rs; Rd ← Rd + Rs

SOLUCIÓN

1) Para direccionar datos en una memoria de 1KB se necesitan 10 líneas de dirección.
Para direccionar instrucciones en una memoria de 8K, se necesitan 13 líneas de dirección.
Los datos son de 8 bits
Para direccionar los registros, como son 16, se necesitan 4 bits.

	17	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Tipo 1	0	0	0	1	x	x	opcode				Rs				Rd			
Tipo 2	0	0	1	opcode			dato								Rd			
Tipo 3	0	1	opcode				dirección de instrucción											
Tipo 4	1	Opcode					dirección de dato								Rd			
Tipo 5	0	0	0	0	x	x	x	opcode			Ri				Rd			

Como se puede ver, las instrucciones deberían ser de 18 bits.

2) ADD Rd, Rs; Rd ← Rd + Rs

FETCH

MAR <- PC
PC <- PC + 1 (*)
MBR <- (MAR)
RI <- MBR

DECODIFICACIÓN

El decodificador interpreta el dato que hemos almacenado en el RI

EJECUCIÓN

Rd <- Rd + Rs (**)

Comentarios:
(*) Si el tamaño de la palabra de la memoria son 32 bits, el PC se incrementa de uno en uno.
(**) Puesto que son registros internos, no se utilizan ni el MBR, ni el MAR.

UNIVERSIDAD CARLOS III DE MADRID (Dpto. de Tecnología Electrónica)
Sist. Dig. Basados en Microprocesador **(Gr. Ing. Telemática)**
6 de mayo de 2014 2º PARCIAL (100 minutos)

PROBLEMA 3 (5.0 puntos): *SOLUCIÓN*

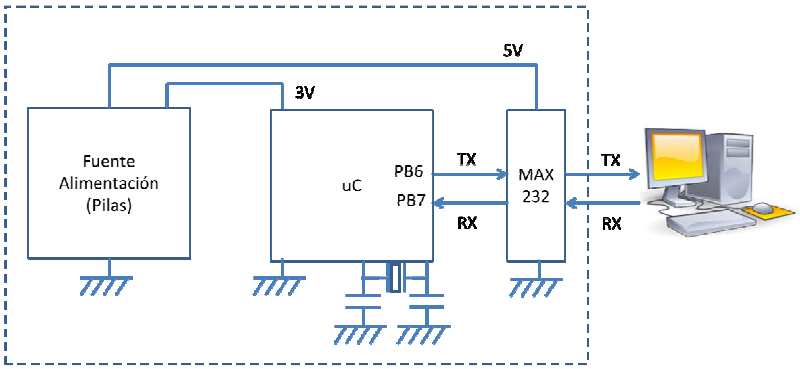
Muchos productos software utilizan "llaves hardware", o lo que es lo mismo, dispositivos hardware que se conectan al ordenador para verificar que el producto software que está utilizando dispone de licencia. Actualmente trabaja usted en una empresa que le ha solicitado la implementación de un prototipo de "llave hardware" para puerto serie, es decir, un dispositivo que conectado al puerto serie de su ordenador verifique que el usuario dispone de una licencia.
La especificación indica que el PC enviará a la "llave hardware" un byte de control con 8 bits de datos, debiendo responder la "llave hardware" con el comando "ACK" en ASCII al recibir dicho comando.
Esto hará que el PC, en caso de no recibir "ACK" detecte que el programa está trabajando sin licencia (no obstante esto no tiene demasiada relevancia para usted dado que sólo le han encargado el diseño de la "llave hardware").
El producto es autónomo, utilizando para su fuente de alimentación basada en pilas, que ofrece dos salidas a 3,3 V y 5 V.

Para la segunda reunión de proyecto, su superior le solicita:

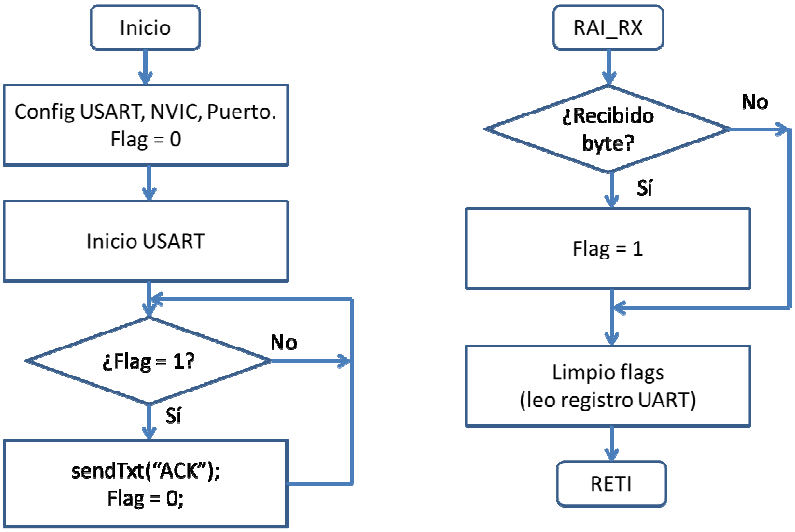
1) Diagrama de bloques del sistema incluyendo todos los dispositivos (alimentación, adaptadores de nivel, osciladores, etc). (0,5 puntos)
2) Diagrama de flujo de la aplicación (cuya recepción funcionará por interrupciones). (1,0 punto)
3) Rutina de configuración de la USART (incluir puertos e interrupciones): `void configUSART(void);` (1,0 punto)
4) Rutina de atención a la interrupción: `void USART1_IRQHandler(void);` (1,0 punto)
5) Rutina de envío de comandos por polling: `void sendTxt(char *dato);` (0,5 puntos)
6) Programa principal: `int main(void);` (1,0 punto)

SOLUCIÓN

1)



2)



3)

```
void configPuetoSerie(){

      // TX en PB6, RX en PB7
      GPIOB->MODER |= (0x01 << (2*7+1));
      GPIOB->MODER &= ~(0x01 << (2*7));
      GPIOB->MODER |= (0x01 << (2*6+1));
      GPIOB->MODER &= ~(0x01 << (2*6));
      GPIOB->AFR[0] = 0x77000000;

      // 9600, 8, N, 1 (fclk = 32MHz)
      USART1->CR1 = 0x0000000C;
      USART1->CR2 = 0x00000000;
      USART1->BRR = 0x00000D05;
      USART1->CR1 |= 0x01 << 13;
      USART1->CR1 |= 0x01 << 5;

      // INT RX
      NVIC->ISER[1] |= (1 << (37-32));

}
```

4)

```
void USART1_IRQHandler(void) {
      if ((USART1->SR & 0x0020)!=0) {
           valor = USART1->DR;
      }
}
```

5)

```
void sendTxt(char *datos){

    while(*datos++){
      while ((USART1->SR & 0x0080)== 0);
            USART1->DR = *datos;
    }

}
```

6)

```
/*
La variable "flag" debe ser global y estar
definida antes del main.
*/

int main(void){

   Init_SDM();
   configPuertoSerie();

   while (1) {

      if(flag){
            sendTxt((char *)"ACK");
            flag = 0;
      }

   }

}
```