

**CIRCUITOS ELECTRÓNICOS DIGITALES**  
**ESCUELA POLITÉCNICA SUPERIOR - UNIVERSIDAD AUTÓNOMA DE MADRID**

**Práctica 4b: Circuitos Basados en Memorias tipo ROM (Tablas de *Look-Up*)**

---

**Problema 1 (5 puntos) Concepto de LUT (*look-up table*).**

Diseñar y construir un circuito que calcule los dos primeros decimales de la parte fraccionaria de una función trigonométrica, para dos rangos de ángulos que varían según el último número de DNI de cada integrante del grupo.

Salida del circuito: Un número de 8 bits que indica las 2 primeras cifras después de la coma del valor de la siguiente función trigonométrica:

Turno Martes:  $\text{sen } \beta$

Turno Miércoles:  $\text{cos } \beta$ .

Turno Jueves:  $\text{tan } \beta$

Turno Viernes:  $\text{cotg } \beta$

Por ejemplo, si  $\text{sen } 30^\circ = 0,500000$ , se pretende que en la dirección 30 de la memoria (0001 1110 = 1E hex) contenga el valor 50 en binario (0011 0010 = 32 hex).

Entrada del Circuito: El ángulo (un número natural de 8 bits) debe entrar a la memoria por las líneas de dirección más bajas A7 A6 A5 A4 A3 A2 A1 A0. Las direcciones que no se usen deben conectarse a 0. La línea A7 se conectará a un interruptor y se utilizará para "paginar" la memoria. Por los interruptores del entrenador se ingresarán los ángulos en binario.

Por simplicidad, en lugar de almacenar todos los valores de una función trigonométrica como ocurre en una calculadora o en un sintetizador de onda, el rango de ángulos se reducirá a dos:

Primer rango:

[Cifra menos significativa del DNI del primer estudiante del grupo]  $\leq \beta \leq$  [Cifra menos significativa del DNI + 10<sup>0</sup>] a pasos de 1<sup>0</sup>.

Es decir, el rango de ángulos resultante será un subconjunto de [0<sup>0</sup>,19<sup>0</sup>].

Segundo rango:

[128<sup>0</sup> + cifra menos significativa del DNI del segundo estudiante del grupo]  $\leq \beta \leq$  [138<sup>0</sup> + cifra menos significativa del DNI] a pasos de 1<sup>0</sup>.

Es decir, el rango de ángulos resultante será un subconjunto de [128<sup>0</sup>,147<sup>0</sup>].

Por lo tanto, cada estudiante debe hacer su propio Intel Hex. Después juntar los records en un solo fichero de texto y finalmente agregar la línea de fin de fichero. Con

una sola grabación de la EEPROM se resuelven ambos casos, pues los datos se graban en diferentes zonas (páginas) de la memoria. Para verificar, la línea A7 servirá para cambiar de página.

### **Problema 2 (5 puntos) Concepto de FSM basada en ROM**

Utilizar el contador 74xx163 y la memoria EPROM para hacer el control de un semáforo simplificado (sin sensor) similar al de la Práctica de FSMs de su turno de laboratorio.

- **Turno Martes:** 2 ciclos verde en cada sentido y un 3er ciclo verde-ámbar
- **Turno Miércoles:** 3 ciclos verde en cada sentido y un 4to ciclo verde-ámbar.
- **Turno Jueves:** 4 ciclos verde en cada sentido y un 5to ciclo verde-ámbar.
- **Turno Viernes:** 5 ciclos verde en cada sentido y un 6to ciclo verde-ámbar.

Invente un mecanismo de “reseteado” del contador para que el semáforo funcione cíclicamente a partir de una señal clk.

---

## Apéndice A: FORMATO INTEL HEX

El primer paso en la grabación de una memoria tipo EEPROM es escribir los datos que se desean introducir en un fichero de texto con un formato compatible con los grabadores de EPROMs que se disponen en el laboratorio. De todos ellos, uno de los más usados es el Intel HEX. Un archivo HEX típico tiene el siguiente aspecto:

```
:10008000AF5F67F0602703E0322CFA92007780C3FD
:1000900089001C6B7EA7CA9200FE10D2AA00477D81
:0B00A00080FA92006F3600C3A00076CB
:00000001FF
```

Los datos se dividen en campos (*records*), cada uno en una línea. Tomando como ejemplo la primera línea, el formato es:

El primer carácter (:) indica el comienzo de un campo.

Los dos caracteres siguientes indican la longitud (número de datos) en bytes del campo como un número hexadecimal de dos cifras. Por ejemplo, este valor es 10 (siempre en hexa) para el primer campo del ejemplo. Es decir, 16 (en decimal) bytes. Usualmente, no se graban más de 16 datos por *record*.

A continuación, los cuatro caracteres siguientes indican la posición inicial de memoria donde se cargarán los datos del campo representada como un número hexadecimal de cuatro cifras (0x0080 en hexa en el ejemplo; o sea, 128 decimal)

Los dos caracteres siguientes indican el tipo de campo, 00 si se trata de datos, 01 para fin de fichero y 02 para dirección extendida (esta última opción escapa del contenido de CEDG).

A continuación vienen los datos (que internamente en la memoria son números binarios de 8 bits); cada uno de ellos representado en el formato Intel Hex como un número hexadecimal de dos cifras (dos caracteres por dato).

Por último, los dos últimos caracteres son el *checksum*. Este *checksum* se calcula de manera que la suma de todos los bytes del campo más el propio *checksum* sea 0x00 (00 en hexadecimal). Sólo se considera el byte menos significativo de la suma.

Si Ud. no ha entendido el párrafo anterior, lo más sencillo es proceder como se indica abajo. Por ejemplo, para la primera línea (*record*) el *checksum* vale FD y se ha obtenido así:

Primer Paso: Se suman todos los bytes del *record*, con excepción del carácter correspondiente a los dos puntos. Es decir:

$$10+00+80+00+AF+5F+67+F0+60+27+03+E0+32+2C+FA+92+00+77+80+C3 = 803$$

Segundo Paso: Se trunca el resultado y se trabaja solo con el byte menos significativo. En este caso, 03. A este número se le halla el complemento a 2. Es decir, se deben cambiar los 1 por 0 y viceversa (complemento a 1) y después sumar 1. En el ejemplo sería:

03 = 0000 0011

Complemento a 1 de 03 = 1111 1100

Complemento a 2 de 03 = 1111 1100 + 1 = 1111 1101

En número resultante, que en hexa se expresa como FD, es justamente el *checksum* del *record*.

El último campo (la última línea) siempre tiene la misma forma, e indica el fin de archivo (campo de tipo 01).