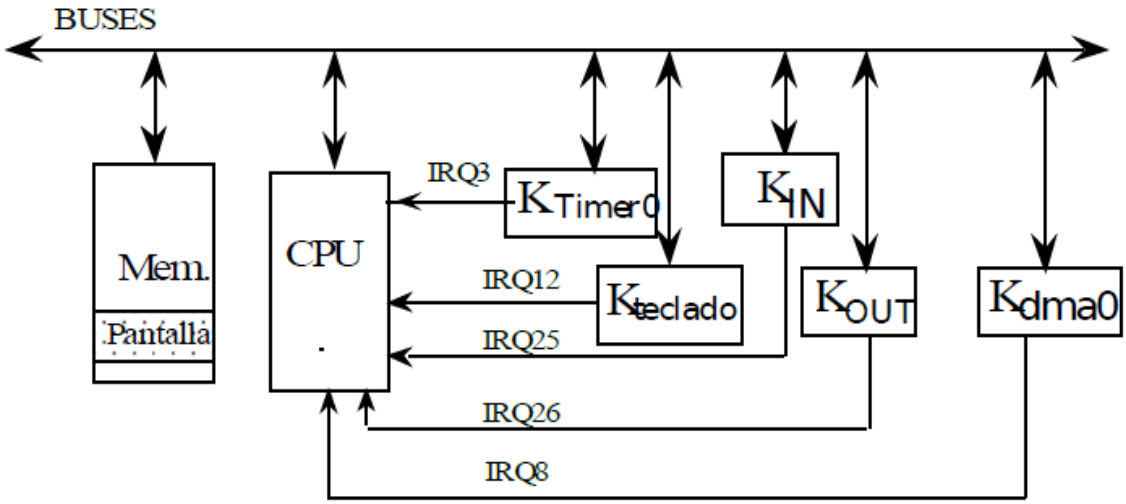
EJECICIO 1

2.- Se quiere realizar de manera automática el control de vehículos en un **parking** céntrico de la ciudad. La estructura hardware del sistema es la que puedes ver en la siguiente figura.



Los controladores de los periféricos que debe controlar el microprocesador son idénticos a los vistos en clase, exceptuando los siguientes:

*** KIN** : es el controlador de un botón situado a la entrada del parking, que es pulsado por el conductor del vehículo que quiere entrar. Dicha pulsación genera una petición de interrupción por la correspondiente línea IRQ (ver figura) proporcionando en su registro de datos (RDAT_KIN) la matrícula del vehículo que se encuentra a la entrada del parking. Cada vez que se acepta una interrupción es necesario realizar una secuencia de STROBE en su registro de control (RCON_KIN).

*** KOUT**: se trata de un lector de tarjetas. Cuando una tarjeta es introducida genera una petición de interrupción por la correspondiente línea IRQ y proporciona en dos registros de datos (RDAT1_KOUT y RDAT2_KOUT) la matrícula del vehículo y la hora a la que entró respectivamente. Cada vez que se acepta una interrupción es necesario realizar una secuencia de STROBE en su registro de control RCON_KOUT).

*** El timer o temporizador, el teclado y el control de las interrupciones son los propios de la NDS, los vistos en clase** y en este sistema el **teclado se sincroniza mediante encuesta.**
El **funcionamiento** del sistema debe ser el siguiente. Cuando un vehículo se dispone a entrar en el parking, su conductor pulsará el correspondiente pulsador situado a la entrada. Como resultado se debe proporcionar al vehículo la correspondiente tarjeta de entrada en la que figurarán tanto la matrícula como la hora de entrada. Para ello suponemos ya implementada la rutina **expedir_billete (horaent,matricula)** a la que hay que proporcionar los dos datos que indican los parámetros: hora de entrada y matrícula del vehículo. A continuación se ha de levantar la barrera de entrada para lo cual se tiene la rutina **levantar_barrera_entrada()**. Dicha barrera permanecerá levantada 3 segundos, tras los cuales se procederá a bajarla mediante la rutina **bajar_barrera_entrada()**. Si el número de vehículos en el parking llega al TOPE admitido, se deben inhibir las interrupciones del pulsador situado a la entrada del parking (volviendo a permitirlas cuando sea posible). En lo que a la hora se refiere, se dispone de la rutina **actualizar()** que actualiza la variable global HORA (consta de segundos, minutos y hora) en un segundo.
Cuando un vehículo se dispone a salir del parking, el cobrador situado a la salida introduce la tarjeta de entrada del vehículo en el correspondiente lector. El cálculo del importe debe ser automático y para realizarlo se cuenta con la rutina ya implementada **calcular (horaent,horasal,&importe)** que además de calcular el importe que debe pagar el conductor, lo visualiza en una pequeña pantalla de varios dígitos y lo devuelve en la variable *importe*. Una vez realizado el cobro, el cobrador pulsará la tecla 'B' y se levantará la barrera de salida mediante la rutina **levantar_barrera_salida()**. Dicha barrera permanecerá levantada 5 segundos, tras los cuales se procederá a bajarla mediante la rutina **bajar_barrera_salida()**.
Se pide:
Escribe en lenguaje algorítmico las rutinas de atención de KIN, KOUT, Ktimer0, así como el programa principal del sistema.

RESOLUCION:

Para la resolución de este ejercicio iremos viendo paso a paso cada estado y explicando en cada caso las operaciones que debería de hacer la subrutina al interrumpir. Antes de empezar con la resolución debo decir las variables globales que habrá en el ejercicio.

- HORA: nos marcara la hora después de haber utilizado previamente la función actualizar() para actualizar la hora y cambiar su valor.
- NumeroCoches: esta variable nos informara del numero de coches que hay dentro del parking.
- Tope: Esta variable es estática y su valor no cambia. Nos indica la cantidad máxima de coches que pueden aparcar en el parking.
- HoraBarrera: esta variables la utilizaremos para guardar la HORA más unos segundos par así levantar o no la barrera de entrada o de salida. (La función de esta variables se aclarará en la explicación de los estados de barrera como posteriormente nombraré y explicaré).
- Matricula: Esta variable global la utilizaremos para guardar en un caso u otro las matricula de los coches con los que se desea realizar las acciones.

ESTADO 1: Normal.

Este primer estado lo he llamado estado "***Normal***" porque es el estado base del ejercicio. Este estado podrá pasar ha varios estado dependiendo de que periféricos lo interrumpan y dependiendo también del espacio de coches que hay en el parking. Por lo tanto esto son los cambios que el estado puede realizar:

```
if (NumeroCoches < Tope  &&  K_IN) { …
```

A)

Si el controlador del periférico del botón K_IN ha solicitado una interrupción y además si la variable NumeroCoches no es igual a la variable Tope se realizarán estas acciones en la subrutina de atención a la interrupción de la señal K_IN:

-Guardaremos la información del registro de control de la señal K_IN, que nos proporciona la matricula del coche que quiere entrar. Para acceder a este registro utilizaremos la función predefinida InPort(dir_regis).

```
matricula= InPort(RDAT_K_IN);
```

-Actualizaremos el valor de la variable HORA para utilizarla posteriormente en la entrega del billete. Para retirar el billete con sus datos correspondientes

utilizaremos la función predefinida expedir_billete(HORA,matricula). Por último haremos el strobe necesario después de leer el registro.

```
Actualizar();
expedir_billete (HORA,matricula);
Strobe(RDAT_K_IN);
```

Después de haber realizado estas acciones la subrutina que trata la interrupción de K_IN nos pasará al estado ***Barrera_Entrada***.

B)

Otra variantes para el estado normal es que la variable NumeroCoches coincida con la variable Tope, por lo tanto ya no entrarían más coches al parking. Para ello he creado un nuevo estado llamado ***Espera*** que esperará a que salga algún coche del parking para que su valor decrezca y pueda entrar algún coche si es que esta esperando alguno. Por lo tanto en este estado no estarán permitidas ningunas interrupciones.

```
If (NumeroCoches = Tope){ …
```

C)

La ultima opción para el estado normal es cuando se activa la señal K_OUT. El controlador de esta interrupción pide interrumpir cuando se ha metido una tarjeta por la maquina de salida del parking para posteriormente sacar el coche.

```
If (K_OUT){...
```

ESTADO 2: Barrera_Entrada.

En este estado se realizan las operaciones y llamadas a funciones necesarias para subir la barrera cuando algún coche quiera entrar al parking. Este es el siguiente paso al estado ***Normal*** y aquí se realizarán estas acciones para llevar a cabo su labor:

-Levantaremos la barrera, para ello utilizaremos la función predeterminada Levantar_barrera_entrada();

-Utilizaremos la variable global HoraBarrera para guardar el valor de la variable HORA(previamente actualizada) más una variable local llamada 3_segundos. Con esto tendremos el valor exacto de la hora en la que la barrera se tendrá que bajar. Por lo tanto podríamos hacerlo de la siguiente manera:

```
HoraBarrera =  HORA + 3_segundos;
```

```
While ( HoraBarrera >= HORA){
        Actualizar();   }
```

-De esta manera cuando hayan pasado lo 3 segundos posteriores a la llamada de elevar la barrera se saldrá del bucle, por lo tanto hay podremos hacer la llamada a bajar a la función predefinida Bajar_barrera_entrada() para que nos baje la barrera.

```
Bajar_barrera_entrada();
```

-Para terminar con las acciones por ultimo la subrutina deberá incrementar el valor de la variable global NumeroCoches porque habrá entrado un coche al parking.

```
NumeroCoches ++;
```

Por lo tanto al final de esta instrucción regresaremos al estado base del ejercicio, estado ***Normal*** directamente.

ESTADO 3: Espera.

Este estado se activará cuando el valor de la variable global NumeroCoches coincida con el de la variable global Tope. En este estado se quedará esperado hasta que un coche salga del parking para que la variable NumeroCoches decrezca y así poder volver al estado normal para que pueda entrar otro coche. Se podría implementar de esta manera:

```
While ( NumeroCoches= Tope){

                }
```

-Entonces así cuando las variable cambie nos saldremos del bucle y podremos volver a continuar volviendo al estado base, ***Normal***.

ESTADO 4: Salida.

Esta es la última opción para el estado normal, cuando se activa la señal K_OUT. En este estado la subrutina de atención a la interrupción deberá de realizar varias acciones para que el coche que estaba en el parking salga. Debe realizar estas acciones:

-Se deben leer los registros de datos del controlador del botón K_OUT que nos dan la información de la matricula del coche que va a salir y la hora en la que el coche a entrado al parking para posteriormente calcular el precio. Los registros son RDAT1_K_OUT, RDAT2_K_OUT respectivamente. Por lo tanto

utilizaremos de nuevo la función predefinida InPort(dir_regis).

```
InPort(RDAT1_K_OUT);
InPort(RDAT2_K_OUT);
```

-A continuación guardaremos en la variable global de matricula la información que nos da el primer registro. También haremos lo mismo con el tiempo que nos proporciona el registro RDAT2_K_OUT que lo guardaremos en una variable local nueva para guardar el valor llamada HoraInicio.

```
Matricula = RDAT1_K_OUT;
HoraInicio = RDAT2_K_OUT;
```

-Luego realizaremos el strobe con el registro del controlador.

```
Strobe(RCON_K_OUT);
```

- Una vez leídos y guardados todos los registros en sus variables procederemos al calculo de el importe. Para ello también deberemos coger la HORA en ese momento para calcular el tiempo que ha estado en el parking aparcado el coche. Para ello utilizaremos nuevamente la función actualizar() y guardaremos el valor de la variable HORA(aunque no es necesario guardarla en otro registro para luego utilizarla, podríamos utilizarla directamente pero lo hago para una mejor comprensión del ejercicio) en una variable local llamada HoraSalida. Por lo tanto con esto ya tendremos todo listo para hacer la llamada a la función predefinida calcular(horaent,horasal,&importe) y que nos haga las operaciones respectivas al calculo del importe(el resultado nos lo devolverá en la variable resultado).

```
Actualizar();
HoraSalida = HORA;
Calcula(HoraInicio,HoraSalida,&importe);
```

-Una vez realizadas estas acciones el operario deberá pulsar el botón B para pasar al estado de Barrera_Salida. Para ello, suponiendo que no haya otro botón aparte del B, utilizaremos la función predefinida MAKE(Código_tecla) que nos dará valor 1 si la tecla ha sido pulsada (MAKE) o 0 si se trata de un BREAK.

```
MAKE(Código_tecla_B);
```

De esta manera al realizar estas acciones pasaremos al 5º Estado, ***Barrera_Salida***.

ESTADO 5: Barrera_Salida.

Este estado será el estado posterior al estado de ***Salida*** y en este estado realizaremos las acciones previas para levantar y bajar la barrera.

-Primero levantaremos la barrera con la función previamente definida

```
Levantar_barrera_salida().
Levantar_barrera_salida();
```

-Después actualizaremos la variable HORA y sumada con la variable local 5_segundos reutilizaremos la variable local HoraBarrera y haremos lo mismo que se ha llevado a cabo en el 2º Estado. Haremos un bucle que vaya actualizando la variable HORA hasta que el valor de HORA sea mayo que el de HoraBarrera para posteriormente llamando a la función predeterminada Bajar_barrera_salida() nos baje la barrera del parking.

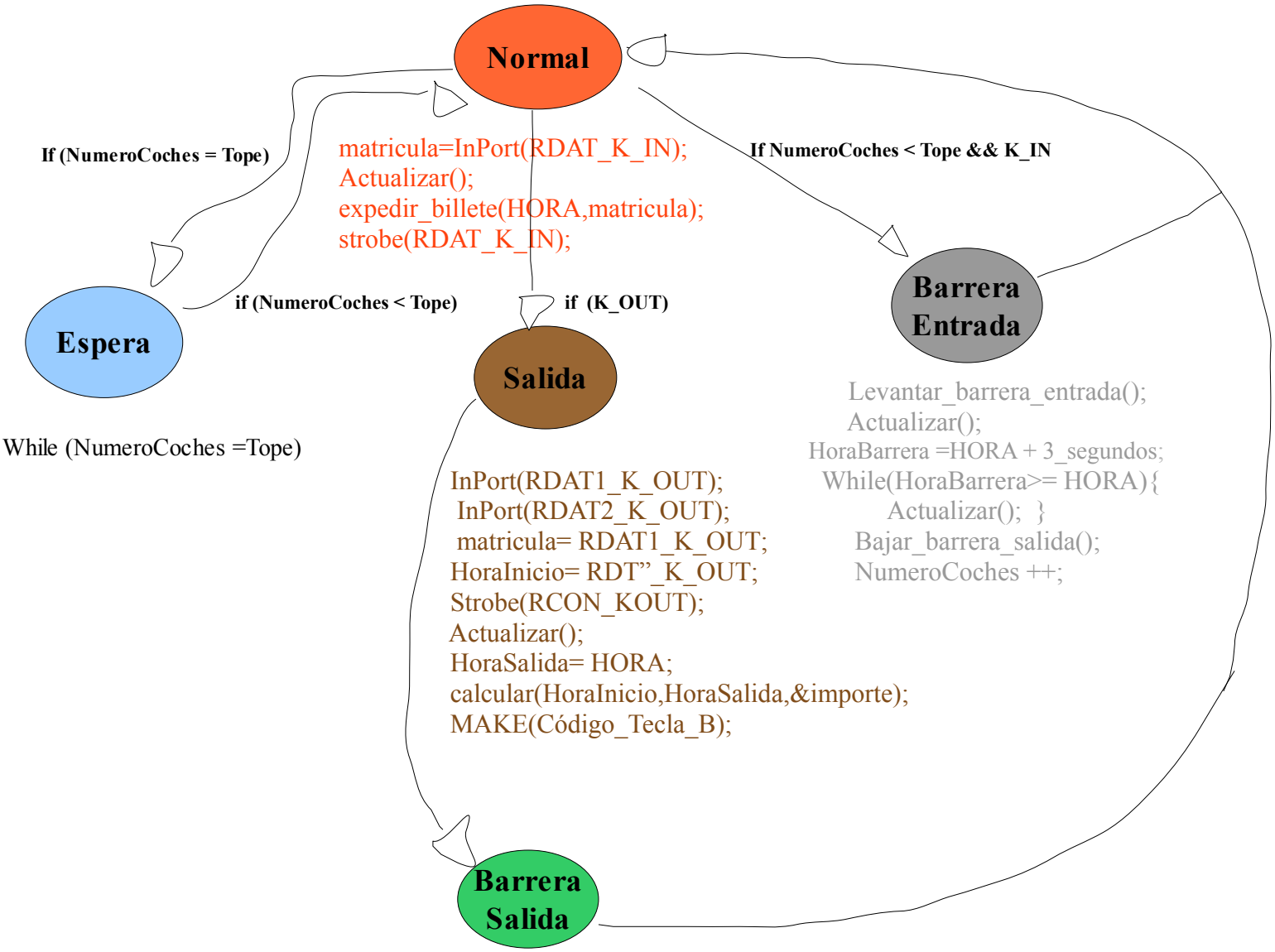
```
Actualizar();
HoraBarrera = HORA + 5_segundos;
While ( HoraBarrera >=HORA) {
        Actualizar();  }
Bajar_barrera_salida();
```

-Por ultimo en este estado reduciremos el valor de la variable global NumeroCoches en 1 porque finalmente ya habrá salido un coche del parking.

```
NumeroCoches – – ;
```

-Después de este estado volveremos otra vez al estado inicial base, ***Normal***.

AUTÓMATA



DANIEL FRANCO BARRANCO