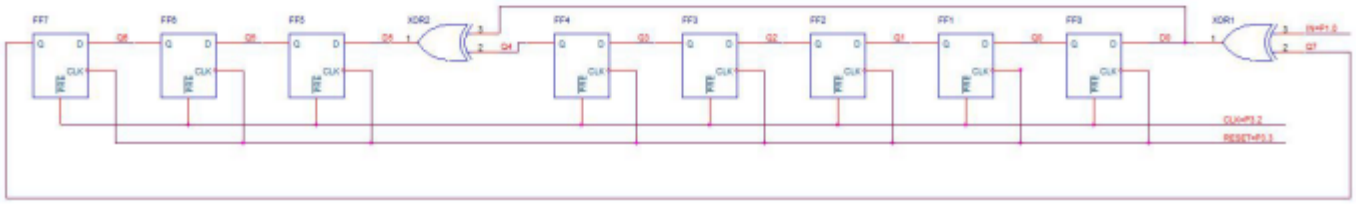
0.2. PROGRAMACIÓN

1 Dado un vector de 25 números en memoria RAM externa ubicado a partir de la dirección Vec1, reescribirlo ordenado de mayor a menor a partir de la dirección Vec2. Los números son enteros sin signo y los vectores no se solapan en ninguna posición.

2 Realizar una rutina que codifique un string ubicado en memoria de código. El dato se envian por el puerto serie usando la función EnviarCaracter y el argumento se pasa en el R7 del banco 0. La codificación es la siguiente: Si es un caracter alfabético, se imprime la letra que está tres posiciones por encima de la original (ej: `'A'-->'D'`) si se llega al fin del alfabeto, se empieza por el principio (`'Y'-->'B','x'-->'a'`). Si es un número se codifica sumandole 17 al ASCII del número (ej:`'0'=48-->48+17=65='A'`). Cualquier otro caracter se invierten los nibbles del valor ASCII (ej: `'<'=60=0x3C-->0xc3=195`).

3 Realizar una rutina que pase un número decimal a BCD empaquetado de un byte. Si no es posible representar el número con un byte BCD, volver de la rutina indicando que hubo error.

4 Programar una rutina que se comporte como el siguiente circuito. Los estados de los flip-flops deben guardarse en el registro ~~13.~~ r20 (Todos los flip-flops actúan en el flanco ascendente del CLK y la entrada (asincrónica) PRE actúa por nivel bajo y deja la salida Q ~~del~~ de cada flip-flop en 1).



5 Se mide una tensión analógica con un conversor AD de 8 bits y una tensión de referencia de 1 Voltio. Programar una rutina que reciba en R0 el valor del AD y devuelva en R4, R3 y R2 las centenas, decenas y unidades del valor de tensión en miliVoltios en formato BCD desempaquetado. (0Voltios se representan como 0000 0000b y 1 Voltio se representa como 1111 1111b)

6 Codificar una rutina que recibe en R0 un valor de temperatura en grados centígrados, y devuelve en R1 el equivalente en grados Fahrenheit. La conversi'on de escalas es $C(\frac{9}{5}) + 32 = F$, donde C es la temperatura en grados centígrados y F es la temperatura en Fahrenheit. Sugerencia: $\frac{9}{5} = 1{,}8 = 0001{,}11001101b$ con un error menor al 0,05 %

7 El método iterativo $x(k+1) = \frac{[x(k)+\frac{a}{x(k)}]}{2}$ permite calcular la raíz cuadrada del valor a. Como valor de arranque se puede tomar $x(k=0) = \frac{a}{2}$. Programar la rutina `SQRT()` que recibe en `R0` un valor a (entero sin signo) y devuelve la raíz cuadrada del mismo en doble precisión, `R7|R6 = SQRT(R0)`. Se debe iterar `N=5` veces. Suponer que se dispone de una rutina de división `DIV()` que calcula esta operación entre dos números de 16 bits. Por ejemplo `R5|R4 = DIV(R5|R4,R3|R2) = R5|R4 / R3|R2`.

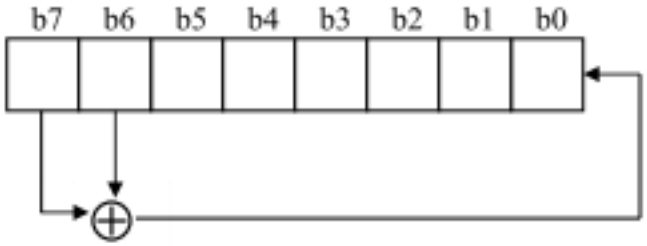
8 Escribir una rutina que divida dos números de dos bytes (número1, número2) que se encuentran en memoria RAM interna y devolver el resultado en una tercer variable de RAM. Definir todos los sectores de memoria necesarios y reservar las variables. Escribir una rutina que lea un teclado matricial conectados en el puerto 1, y que ejecute una función según el número presionado. Las funciones a ejecutar están definidas en una tabla en ROM de la siguiente forma:

```
tabla_funcs: DB low(dir_func1), high(dir_func1), low(dir_func2), ...
```

9 Usando la subrutina `RAND_8` del ejercicio ~~anterior,~~ 10 escribir el código para inicializar ~~el~~ un vector VAR2 en memoria RAM ~~del ejercicio 1~~, con números aleatorios de la siguiente forma: Tomar para R0 el último valor calculado en R2 o 0x78 si es el primero. Para R1 tomar el actual valor del elemento del vector si no es nulo o 0x8F en caso contrario. Invocar a `RAND_8` y asignar como nuevo valor del elemento el de R2. Proceder así hasta completar los 128 elementos del arreglo.

Calcular además la esperanza del vector, E(x)= $\sum x_i / n$ y dejar el valor en R3.

10 Una forma de generar numeros pseudoaleatorios es mediante registros de desplazamiento realimentados. Uno de los mas simples es el siguiente:



donde b0 a b7 son los 8 bits de un registro. En cada ciclo de reloj se desplaza su contenido un bit hacia la izquierda, tal que $b6_{viejo}$ pasa a ser $b7_{nuevo}$, $b5_{viejo}$ pasa a ser $b6_{nuevo}$, etc.; además, el contenido de $b7_{viejo}$ se pierde, y el contenido de $b0_{nuevo}$ es la or-exclusiva entre $b7_{viejo}$ y $b6_{viejo}$.

Se pide programar la subrutina `RAND_8` que implemente este circuito, la cual recibe en R0 el valor inicial del registro, en R1 la cantidad de ciclos de reloj a iterar, y devuelve en R2 el resultado. Si por ejemplo, se recibe R0=80h y R1=0Ah; la subrutina pasa por los estados indicados en la tabla, devolviendo R2=06h.

Valor inicial	80h	10000000 b
1º ciclo	01h	00000001 b
2º ciclo	02h	00000010 b
3º ciclo	04h	00000100 b
4º ciclo	08h	00001000 b
5º ciclo	10h	00010000 b
6º ciclo	20h	00100000 b
7º ciclo	40h	01000000 b
8º ciclo	81h	10000001 b
9º ciclo	03h	00000011 b
10º ciclo (valor final)	06h	00000110 b