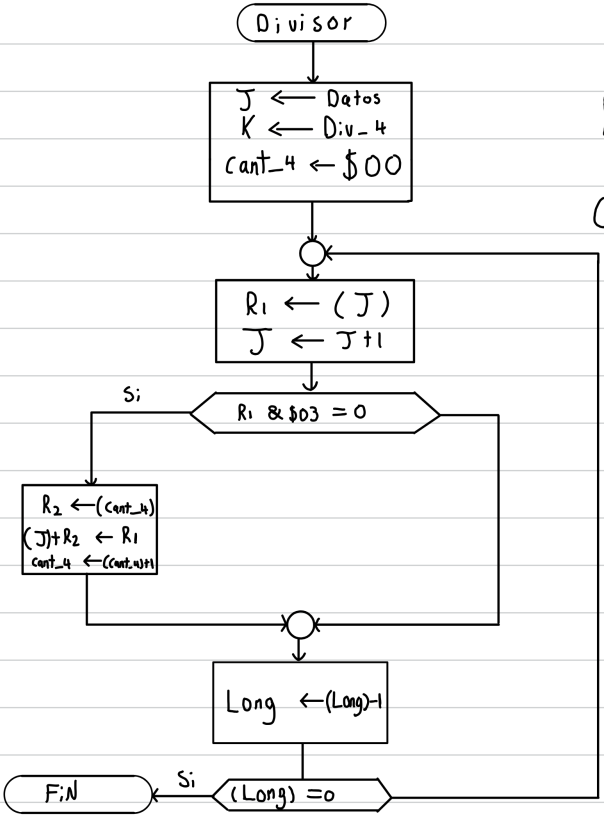
Tarea 2 Microprocesadores IE 0623

Bryan Cortés Espínola C22422

Problema #1. (30 pts)

Considere que se tiene una tabla ubicada en la dirección *Datos* que contiene números de 1 byte, *con signo* y cuyo tamaño esta almacenado en la variable *Long*, donde *Long* contiene un número menor que 255 pero mayor que cero. Se debe diseñar un programa, llamado Divisor, para encontrar todos los números en la tabla que son divisibles por 4 y copiarlos a un arreglo llamado *Div_4*. Adicionalmente el programa debe calcular la cantidad de números divisibles por 4 encontrados y almacenarlo en la variable *Cant_4*. En el diseño el índice J apunta a *Datos* y el índice K apunta a *Div_4*. Durante la ejecución del programa el índice K no debe modificarse. El diseño debe considerar el caso donde todos los *Datos* o ninguno de ellos es divisible por 4.



Estructuras de datos

Datos: Variable tipo word, Contiene la direccion de los datos a procesar.

Div_4: Variable tipo word, Contiene la direccion donde se guardan los datos divisibles pa cuatro

Cant_4: Variable tipo byte, Guarda la cantidad de datos divisibles pa cuatro

Long: Variable tipo byte, Contiene la cantidad de datos a procesar

Notas:

Se Conoce si es divisible si los 2 bit menos significativos es cero, por ende se aplica una mascara con and para sale si son cero

Problema #2. (40 pts)

Se tienen dos tablas, la primera de ellas se encuentra en la posición *Datos* y contiene números con signo en el intervalo [-127,+127]. La segunda tabla contiene máscaras sin signo menores que 254 y se encuentra en la dirección *Mascaras*. Ambas tablas son de tamaño variable menor de 255 pero mayor que 1. El último valor en *Datos* será siempre $80 y el último valor en *Mascaras* será $FE, siendo estos valores los indicadores de fin de tabla. Las tablas, en general, no tienen el mismo tamaño. Se debe diseñar un programa llamado Selector que debe realizar la XOR de los números con las máscaras, el último número con la primera máscara, el penúltimo número con la segunda máscara

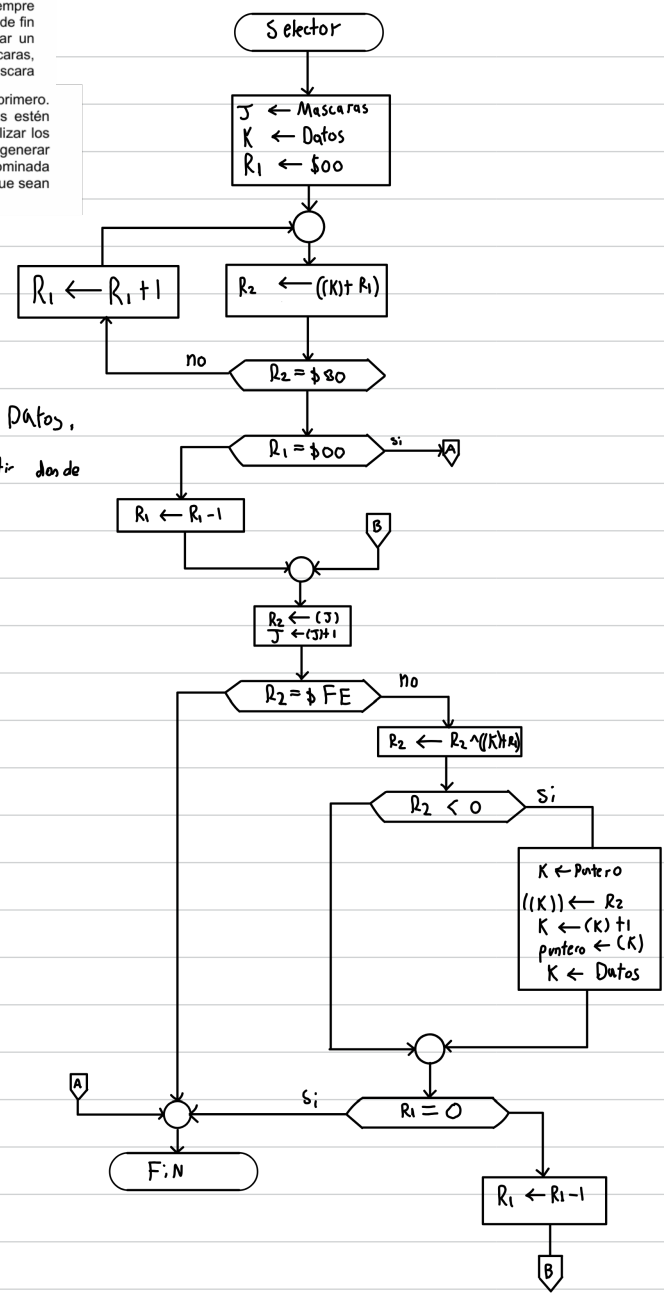
y así sucesivamente hasta procesar todos los datos de la tabla que se termine primero. El diseño del programa debe considerar el caso en que una o ambas tablas estén vacías, es decir, que solo contengan el indicador de fin de tabla. Se deben utilizar los índices J y K para barrer las tablas y además el índice K también se utiliza para generar un arreglo de resultados cuya dirección se encuentra en una variable denominada *Puntero*. En el arreglo de resultados se colocarán los resultados de las XORs que sean números negativos.

Estructuras de dato

Mascaras: Tipo word, Contiene la direccion de la tabla de mascaras.

Datos: Tipo Word, Contiene la direccion de los Datos,

Puntero: Tipo word Contiene la direccion a partir donde de guardan los resultados negativos.



Problema #3. (30 pts)

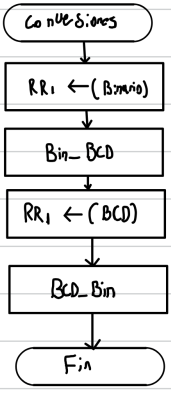
Diseñe y codifique en ensamblador del S12 un programa llamado CONVERSIONES que incluye la implementación de dos algoritmos denominados BIN-BCD y BCD-BIN.

a. Algoritmo BIN-BCD: Este algoritmo toma un número de 12 bits almacenado en el acumulador D y lo convierte a BCD utilizando el algoritmo XS3. El algoritmo debe colocar el resultado en la variable Num_BCD ubicada en la memoria a partir de la posición $1010.
b. Algoritmo BCD-BIN: Este algoritmo realiza la conversión de un número BCD a Binario, utilizando el método de multiplicación de décadas y suma. El número en BCD es menor o igual a 9999 y está ubicado en el acumulador D. El algortimo debe guardar el resultado en las posiciones de memoria Num_BIN ubicadas a partir de la dirección $1020.

El programa principal debe crear los valores a convertir como constantes. El valor binario estará en la constante Binario ubicada en las posiciones $1000-$1001, en tanto el valor BCD a convertir estará en la variable BCD ubicada en las posiciones $1002-$1003.

En el programa CONVERSIONES primero se copia el valor de Binario en el acumulador D y ejecuta el código del algoritmo BIN_BCD, luego el programa copia el valor de BCD en el acumulador D y se ejecuta el código del algoritmo BCD-BIN. Realice las pruebas pertinentes a su programa para garantizar que el programa satisface los requerimientos establecidos. Ubique el programa a partir de la posición $2000.

Diagrama de flujo de conversiones

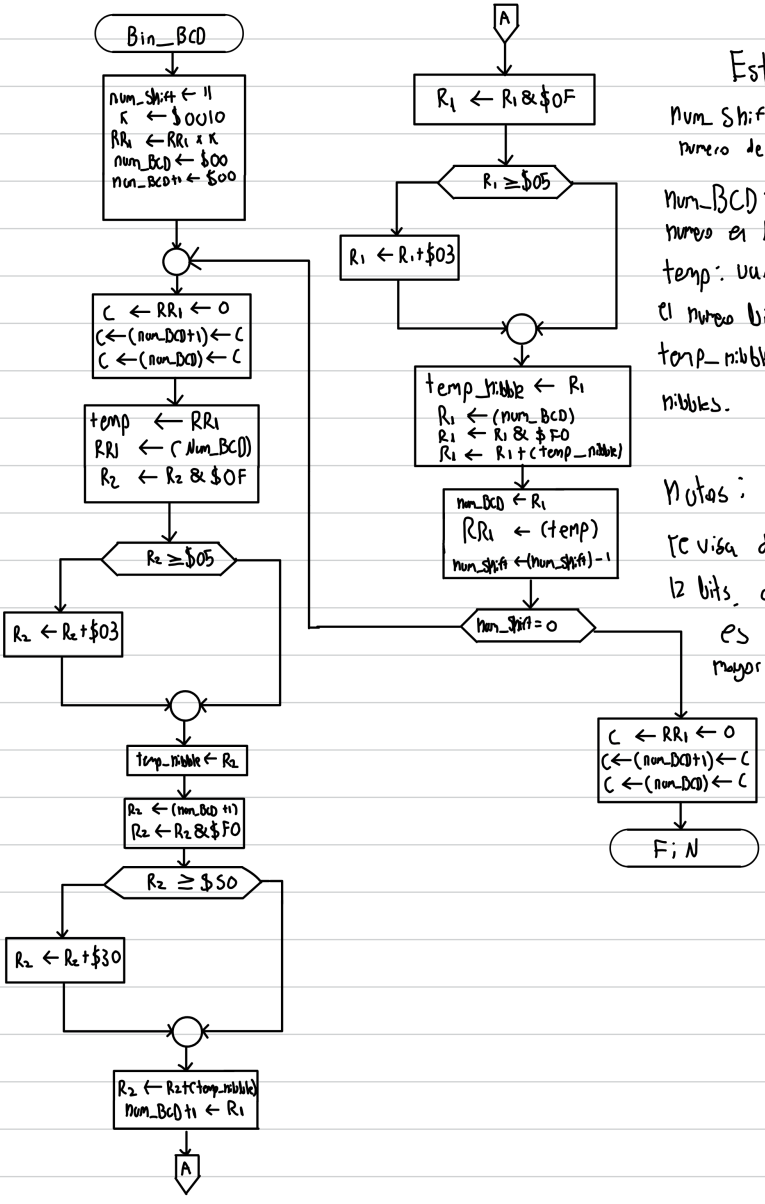


Estructura de Datos

Binario: tipo word, contiene el numero binario de 12 bits a convertir a BCD

BCD: tipo word, contiene el numero en BCD a convertir a binario

Diagrama de flujo del algoritmo Bin_BCD



Estructura de Datos

num_Shift: Variable tipo byte, contine el numero de desplazamiento que se deben realizar

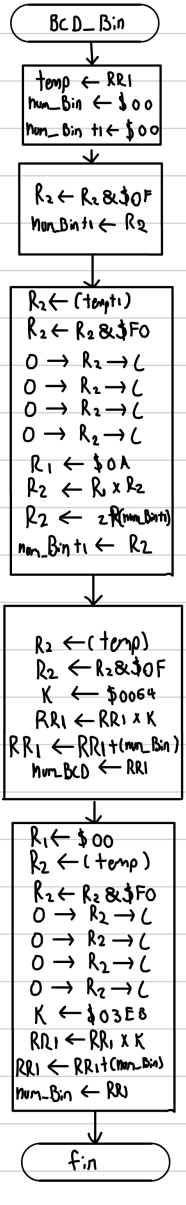
num_BCD: tipo word, donde se guardara el numero en BCD resultante

temp: variable temporal tipo word, guarda el numero binario desplazado en las iteraciones

temp_nibble: Variable tipo byte para guardar nibbles.

Notas: El cuarto nibble no se revisa debido a que se utilizan numeros de 12 bits, donde el numero mayor, que se puede representar es 4095, por ende el ultimo nibble nunca es mayor o igual que 5.

Diagrama de flujo del algoritmo BCD_Bin



Estructura de datos

Num_Bin: tipo word, se guarda el resultado de la conversion BCD-Bin

temp: variable tipo word, se guarda temporalmente el numero BCD a convertir