Danna Guevara Quesada C23562

UNIVERSIDAD DE COSTA RICA
ESCUELA DE INGENIERÍA ELÉCTRICA
MICROPROCESADORES
IE0623

EIE
Escuela de
Ingeniería Eléctrica

TAREA #2

Para cada uno de los siguientes problemas debe utilizar el simulador para hacer todas las pruebas que le permitan validar que los programas cumplen con las funcionalidades requeridas. La calificación de la tarea incluirá, entre otras cosas, dicha validación y la persona estudiante debe verificar que su solución las cumple. En todos los casos se evaluará el cumplimiento de las funcionalidades y la eficiencia de código de los programas.

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.

a. Haga el diseño del programa Divisor que satisfaga los requerimientos funcionales indicados. Únicamente debe utilizar las estructuras de datos indicadas. Su diseño no debe utilizar la pila. Se evaluará eficiencia y eficacia del algoritmo diseñado.
b. Codifique su diseño en lenguaje ensamblador del S12. El programa debe iniciar en $2000. Las direcciones para *Datos* y *Div_4* son $1100, $1200, respectivamente. Además las variables *Long* y *Cant_4* se deben ubicar en las posiciones $1000 y $1001 respectivamente. Ensamble el programa y compruebe que satisface los requerimientos funcionales para un conjunto de datos de su elección. La tabla de Datos debe ser generada en tiempo de ensamblado.

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.

a. Diseñe el programa Selector que cumpla con los requerimientos funcionales descritos. **Unicamente** debe utilizar las estructuras de datos indicadas. Se evaluará eficiencia y eficacia del algoritmo diseñado.
b. Codifique el programa en lenguaje ensamblador de S12 considerando que la dirección del *Puntero* es $1000 colocando el arreglo de resultados a partir de una dirección de su conveniencia. La dirección de *Datos* es $1050, la de *Mascaras* es $1150. El programa debe ubicarse a partir de la posición $2000. El código debe crear las tablas *Datos* y *Mascaras* en tiempo de ensamblado.
c. Ensamble el programa y genere el archivo .S19. Pruebe el programa para unas tablas de su elección. Verifique la correcta operación de su solución implementada utilizando el simulador. Para efectos de la calificación se utilizará un protocolo de pruebas que valide el cumplimiento de los requerimientos funcionales indicados.

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.

Prof. Geovanny Delgado

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.

Remita el **código fuente únicamente** (Archivo .asm) de cada uno de sus programas así como el documento pdf con los diseños, explicaciones, notas de cálculo, etc. El formato de cada archivo de solución debe ser SUNOMBRE_Pk_T2.asm, donde k es el número del problema de la tarea (1,2,3), puede remitir todos estos archivos de manera comprimida en un solo archivo .zip y el documento pdf debe nombrarse SUNOMBRE_T2.pdf.

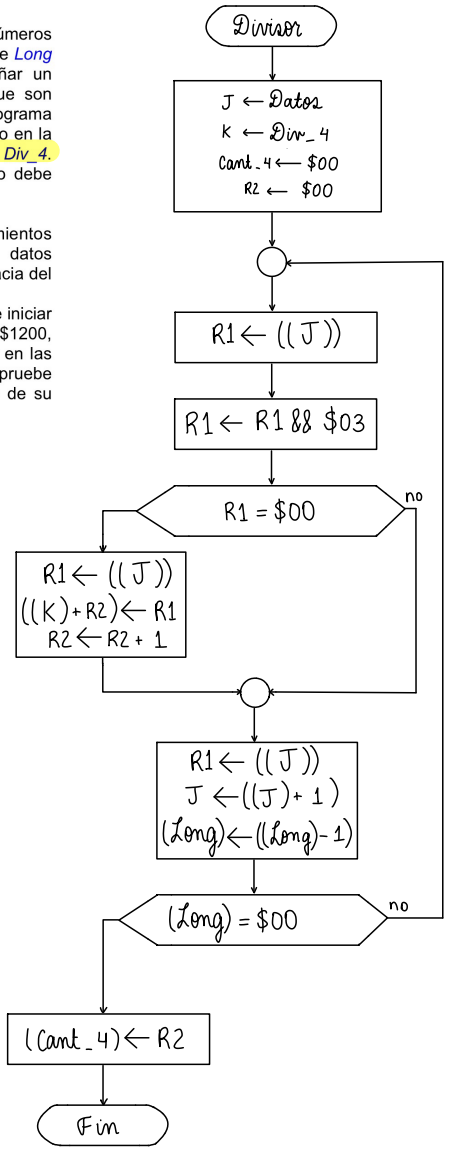
Problema #1. (30 pts)

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.

a. Haga el diseño del programa Divisor que satisfaga los requerimientos funcionales indicados. Únicamente debe utilizar las estructuras de datos indicadas. Su diseño no debe utilizar la pila. Se evaluará eficiencia y eficacia del algoritmo diseñado.
b. Codifique su diseño en lenguaje ensamblador del S12. El programa debe iniciar en $2000. Las direcciones para *Datos* y *Div_4* son $1100, $1200, respectivamente. Además las variables *Long* y *Cant_4* se deben ubicar en las posiciones $1000 y $1001 respectivamente. Ensamble el programa y compruebe que satisface los requerimientos funcionales para un conjunto de datos de su elección. La tabla de Datos debe ser generada en tiempo de ensamblado.

Estructuras de datos:

Datos: arreglo de un byte
Div_4: arreglo que guarda los números divisibles
Cant_4: variable con cantidad de números divisibles
Long: variable que contiene el tamaño del arreglo Datos



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.

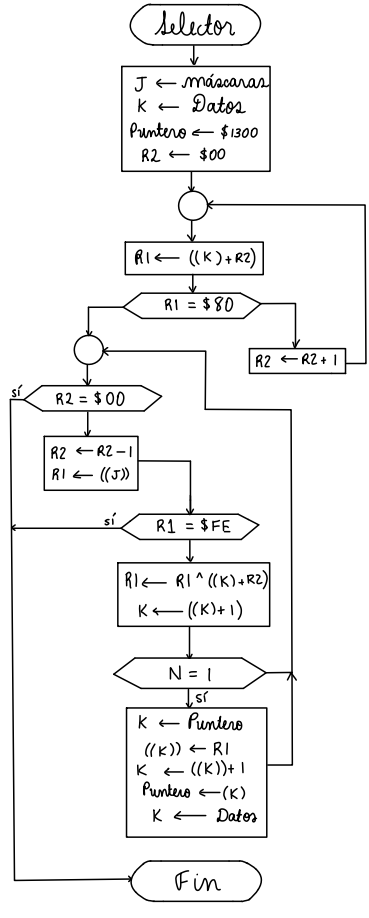
a. Diseñe el programa Selector que cumpla con los requerimientos funcionales descritos. **Unicamente** debe utilizar las estructuras de datos indicadas. Se evaluará eficiencia y eficacia del algoritmo diseñado.
b. Codifique el programa en lenguaje ensamblador de S12 considerando que la dirección del *Puntero* es $1000 colocando el arreglo de resultados a partir de una dirección de su conveniencia. La dirección de *Datos* es $1050, la de *Mascaras* es $1150. El programa debe ubicarse a partir de la posición $2000. El código debe crear las tablas *Datos* y *Mascaras* en tiempo de ensamblado.
c. Ensamble el programa y genere el archivo .S19. Pruebe el programa para unas tablas de su elección. Verifique la correcta operación de su solución implementada utilizando el simulador. Para efectos de la calificación se utilizará un protocolo de pruebas que valide el cumplimiento de los requerimientos funcionales indicados.

Estructuras de datos:

Datos: tabla de números con signo de 1 byte.

Máscaras: tabla de máscaras sin signo de 1 byte

Puntero: variable que coloca los resultados negativos de la XOR



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.

Estructuras de datos:

BIN: constante tipo byte
BCD: variable con un número BCD tipo word.
BCD_L, BCD_H, TEMP, LOW: variables temporales tipo byte
CONT: contador de 12 desplazamientos
NUM_BCD: variable para guarda el resultado
NUM_BIN: posicion de memoria para guardar resultado