Universidad de Costa Rica

Microprocesadores

Tarea 03

Yoel Moya Carmona - B75262

10 de Octubre de 2023

Índice

1. Descripción del diseño

Todos los diagramas de flujo realizados para cada uno de los problemas se encuentran en la sección 2 de anexos.

1.1. Sub rutina GET CANT

El diagrama de flujo de esta sub rutina se encuentra en la sección 2.1.

Se dio uso de las sub rutinas GET CHAR, PUT CHAR, y PRINTF, del debugger, para obtener datos del usuario. Estos fueron utilizados para obtener datos del usuario, mostrar los datos validos ingresados, y comunicar instrucciones con el mismo.

Como los números validos son únicamente valores entre 1 y 25 (incluyendo estos) se utilizo el despliegue en pantalla de las teclas presionadas para mostrar únicamente los valores validos que se encuentran dentro del rango especificado

Como el primer dato ingresado es siempre una decena, se prueban las únicas decenas validas que son 0, 1, y 2. No se pasan a binario, ya que son solo tres valores, lo que abre la posibilidad de comparar directamente la tecla presionada con su equivalente ASCII.

Una vez que la tecla de las decenas es valida, esta se pasa a su expresión en BCD, y se multiplica por 10 ($0A$) lo que resulta en su expresión en binario, con el objetivo de realizar una implementación pequeña del algoritmo de multiplicación de décadas y suma.

Esta multiplicación no es necesaria para las unidades, ya que, al ser pasadas a BCD, estas se encuentran en su expresión binaria, por lo que se procede a realizar la suma y comparar el valor final.

Esta suma se mantiene en el acumulador, y se sigue esperando que se presione una tecla hasta que esta sea valida. Si esto se cumple, se procede a guardar la expresión en la variable especificada. Para mostrar la tecla presionada, este valor se transfiere al acumulador A (Ya que todo el proceso de registro de teclas se realiza en esta parte con el acumulador B), y se muestra en pantalla.

1.2. Sub rutina ASCII BIN

El diagrama de flujo de esta sub rutina se encuentra en la sección 2.2.

Se maneja el puntero de pila de la siguiente manera: En la sección de código principal, se abre el espacio que espera que se reciba la dirección de retorno de la subrutina, se ingresan los valores requeridos por la misma, y se restablece el puntero al inicio de la misma. Esto resulta en que la dirección de retorno se guarde en las partes mas altas de las direcciones de memoria de la pila, ya que esta va a crecer dentro de la sub rutina al guardar un offset utilizado para barrer el arreglo de datos binarios al ser guardados.

Se modifica el puntero en cada ocasión que se necesite extraer un dato, y el momento que este deja de ser usado, se guarda de nuevo en la pila para mantener orden con el puntero.

El offset que barre la tabla de datos, extrae cada posición de memoria de manera individual, por lo que esto permite la utilización de los acumuladores de 8 bits.

Una vez que finaliza se utiliza la implementación de multiplicación de décadas y suma (Luego de transformar los números ASCII a BCD), el puntero de pila se restablece a la dirección de retorno, y una vez que se regresa al programa principal, la pila se encuentra completamente utilizable sin necesidad de restablecer el puntero de pila en este bloque de código.

1.3. Sub rutina MOVER

El diagrama de flujo de esta sub rutina se encuentra en la sección 2.3.

Para separar los valores, se aprovecha de que el nibble mas significativo y el menos significativo se pueden extraer de manera consecutiva sin tener que restablecer el valor del dato en binario, esto porque la palabra que los contiene se encuentra dividida entre ambos acumuladores de 8 bits.

Se utilizan mascaras AND con el valor $\$0F$, para aislar estos dos nibbles, y se procede a guardarlos en sus arreglos respectivos. Se utiliza el índice Y como variable temporal para sostener el dato del cual se extraen los nibbles, para evitar realizar otro acceso a memoria.

El nibble del medio se extrae rotando cuatro veces hacia la derecha con ceros, esto debido a que toma una cantidad mayor de instrucciones realizar la mascara y desplazar luego hacia la parte baja del acumulador, y una cantidad similar el realizar la división por 4 y acomodar todos los punteros nuevamente. Realizándolo de esta manera, el nibble se encuentra aislado y listo para ser guardado en su respectivo arreglo.

1.4. Sub rutina IMPRIMIR

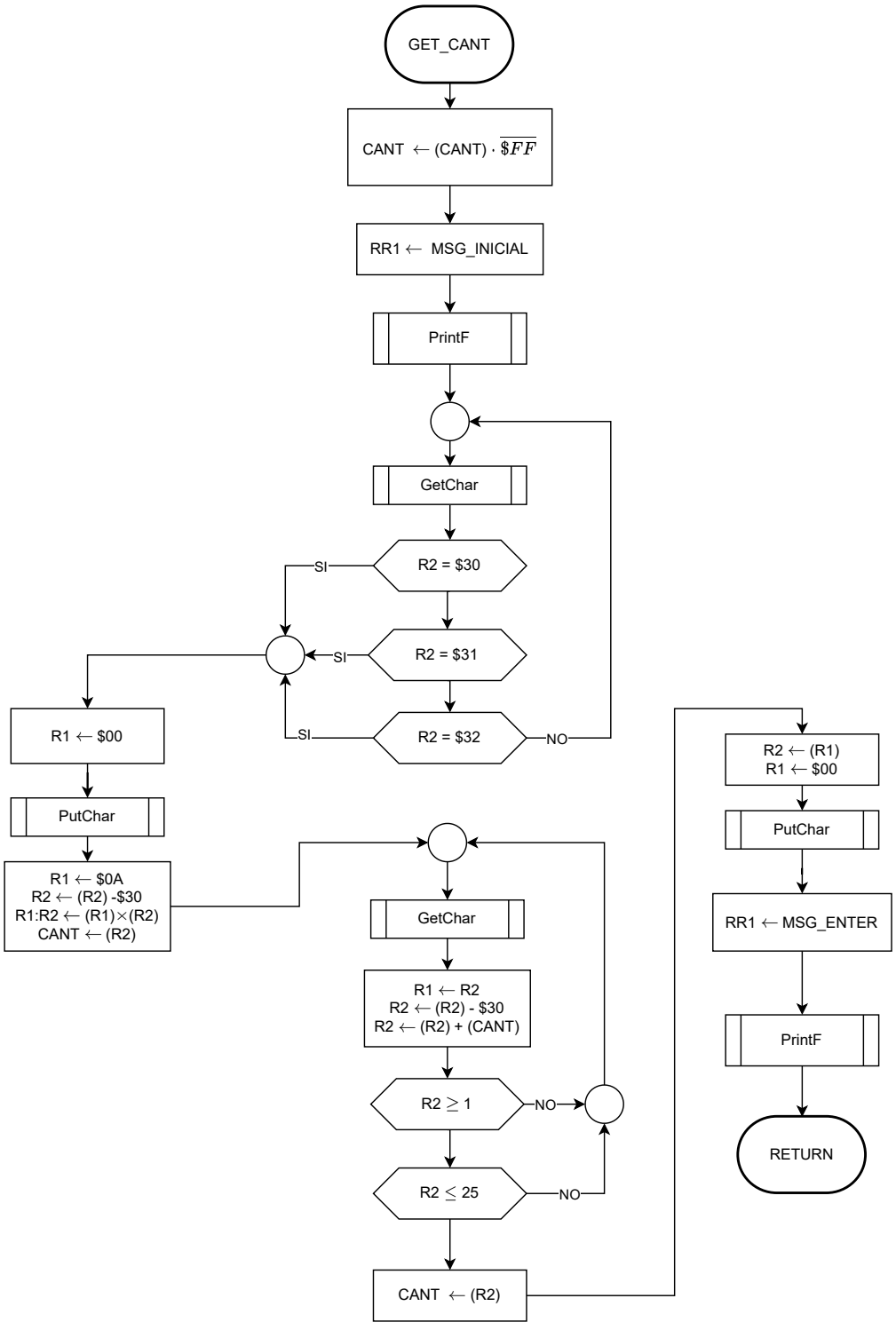
El diagrama de flujo de esta sub rutina se encuentra en la sección 2.4. Sin embargo, este diagrama se encuentra desactualizado de su implementación en el código.

La implementación inicial pretendía realizar un mismo ciclo para imprimir todos los valores de todos los arreglos, realizando un cambio de puntero, sin embargo no se logro realizar esta implementación, y se procedió con una manera mucho menos eficiente pero funcional donde se realiza un ciclo individual para cada uno de los arreglos.

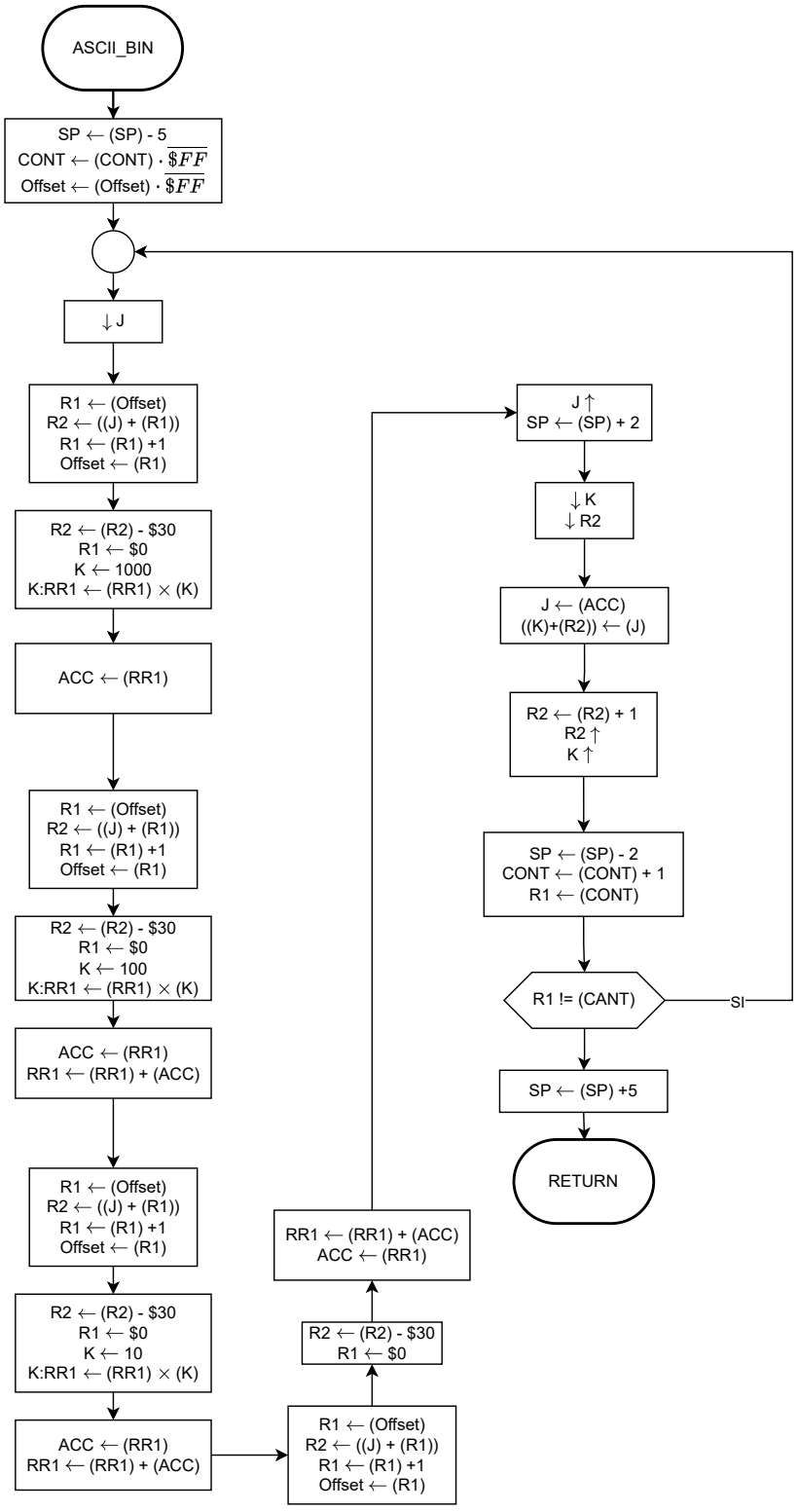
Se prendía inicialmente mantener el offset que itera sobre los arreglos en la pila. Sin embargo esta implementación resulto problemática ya que el puntero de pila se estaba perdiendo. Se opto por utilizar la variable CONT, la cual después de ser desplegada en pantalla, esta no es utilizada mas, como variable temporal que sostiene el offset de cada uno de los arreglos. Esta variable debe ser limpiada antes de iniciar la iteración sobre un nuevo arreglo.

2. Anexos

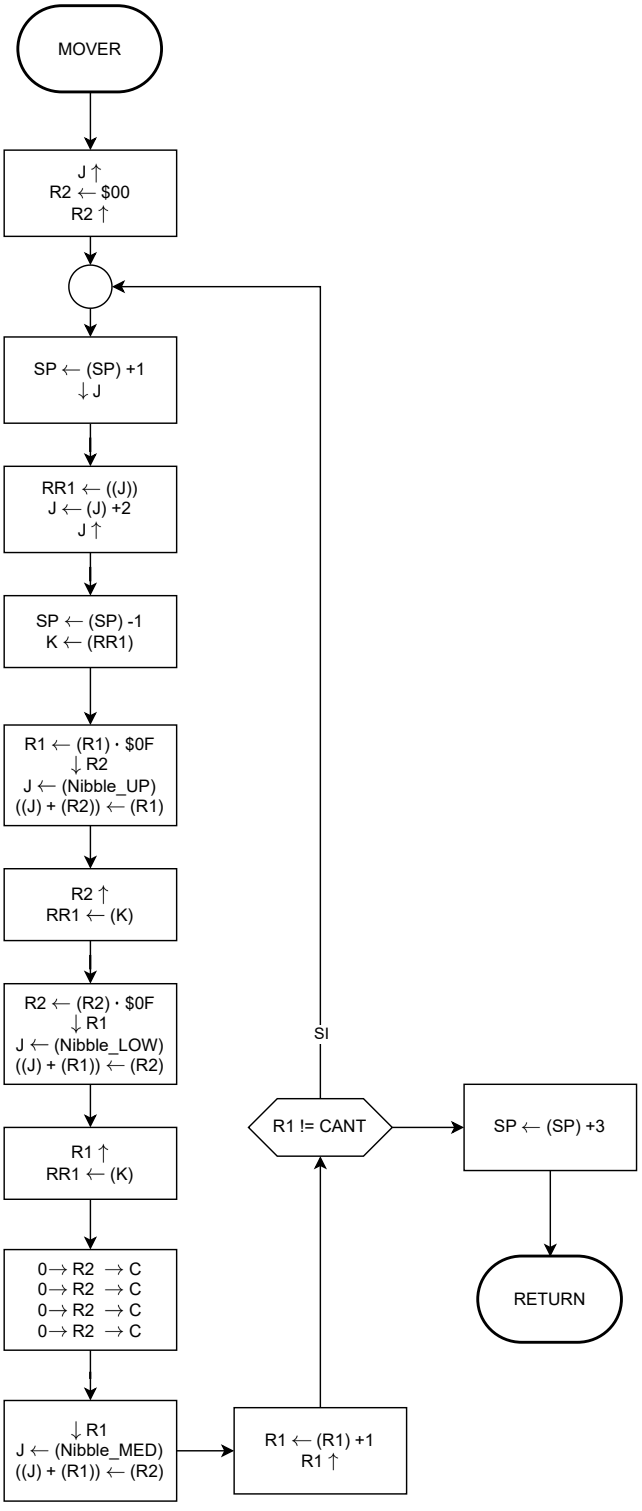
2.1. Diagrama GET CANT



2.2. Diagrama ASCII BIN



2.3. Diagrama MOVER



2.4. Diagrama IMPRIMR