López Chávez Anel Jesali 2163032140
Ramírez Calderón Monserrat Valeria 2163031876
Rodríguez Hernández Luis 2163032177

PRÁCTICA 1. Apertura de periféricos E/S en ARM.

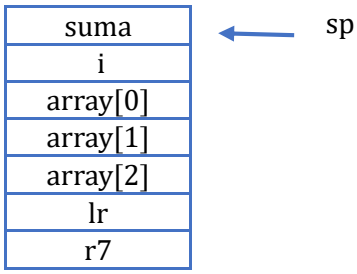
Resumen.

Esta práctica tiene como objetivo familiarizarse con los periféricos de E/S en el lenguaje Ensamblador ARM específicamente. Por lo cual, se busca que mediante la entrada estándar de la terminal de ejecución, se realice el ingreso de los datos requeridos para el manejo del arreglo, así como la salida del valor de la función para la cual es requerido. Además de eso, se busca el manejo de la memoria adecuada para las variables, utilizando la convención de almacenarla en múltiplos de 8 bytes, dependiendo de la cantidad de variables que se van a utilizar.

Desarrollo.

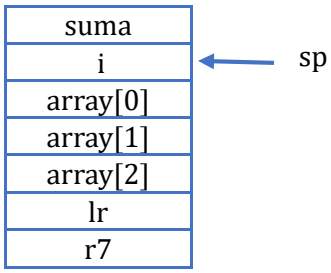
El desarrollo de la práctica se realizará en puntos específicos, que se describirán a continuación.

1. Definición del marco de la memoria que se utilizará. Comenzando con la definición del Prólogo, el cual se encarga de reservar la memoria suficiente para las variables del programa. Se actualiza el valor del sp (Stack Pointer), el cual apunta a la cima de la pila. Por lo que el inicio de la función main se vería de la siguiente manera.



2. Se tiene una función llamada “leer”, la cual permitirá el ingreso de los valores del arreglo y llenado del mismo mediante un bucle for; todo esto por medio de la entrada estándar.

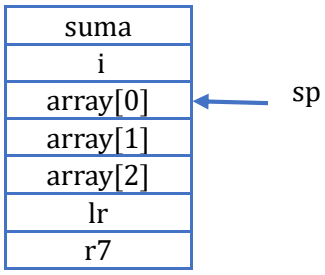
   Tenemos una función leer en la cual ponemos en 0 a los registros, posteriormente realizamos un ciclo for con 3 iteraciones (por el tamaño del arreglo). En el cual llamará en cada una de sus iteraciones a la función convertir.



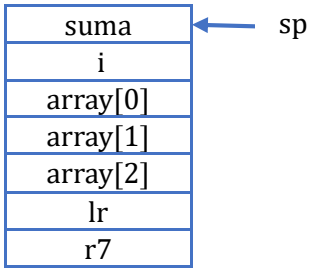
López Chávez Anel Jesali 2163032140
Ramírez Calderón Monserrat Valeria 2163031876
Rodríguez Hernández Luis 2163032177

3. Se tendrá la función convertir, la cual se encargará de convertir las entradas ASCII en enteros, por convención. Utilizando así, la función "atoi".

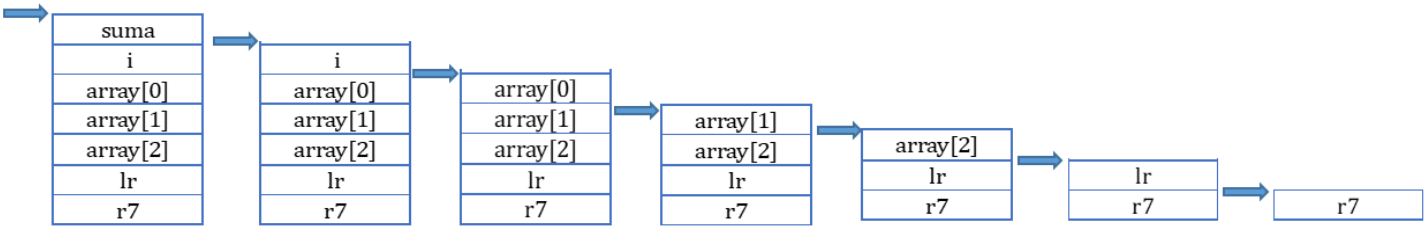
   La función será llamada por leer, donde convertir realizará la conversión de ascii a entero de manera que cuando el número ingresado por terminal sea >48 & <56 (que son los números del 0 al 9 en ascii) realizará la conversión y colocará en las direcciones de memoria del arreglo donde le corresponde.



4. Suma:
   Con los datos previamente cargados en la pila realizaremos la suma del arreglo y por ultimo lo guardará en la variable suma.



5. Por último, se tiene al epílogo que es el código de salida encargado de liberar la memoria que se ocupó durante el proceso del programa. Le regresa el mando al sp.



López Chávez Anel Jesali 2163032140
Ramírez Calderón Monserrat Valeria 2163031876
Rodríguez Hernández Luis 2163032177

Conclusiones.

Se puede concluir que, el manejo del programa a nivel ensamblador es de gran ayuda debido a que se tiene un mejor control sobre la memoria utilizada, es decir, sobre la pila y sobre el manejo de las funciones que se utilizan.

Es importante destacar que, no había que perder la noción de la reserva de memoria ya que la convención establecida es distinta a la normalmente utilizada, por lo que, si se reservaba una menor cantidad de memoria era posible que no se pudiera hacer correctamente el almacenamiento de las variables que se utilizaron en el programa. Por otro lado, si se reservaba una mayor cantidad, no generaba algún problema la pérdida de los bytes no utilizados.