
	<p style="text-align: center;"><b>UNIVERSIDAD DE COSTA RICA</b> <b>ESCUELA DE INGENIERÍA ELÉCTRICA</b></p>	
<p style="text-align: center;"><b>IE-623 MICROPROCESADORES</b> <b>Prof. Geovanny Delgado</b></p>		

### TAREA #3 SUBROUTINAS DEL DEBUG 12

#### Problema:

Se tiene una tabla de datos de 1 byte sin signo, cuya longitud es definida por la constante LONG, ubicados a partir de la posición DATOS. Además se tiene otra tabla llamada CUAD que contiene los números que tienen raíz cuadrada entera en el intervalo de los posibles valores de DATOS (4,9,16,25,36...) Se desea crear un arreglo llamado ENTERO que contendrá la raíz cuadrada de CANT números de DATOS que estén en CUAD. De esta manera si CANT=5, el programa deberá buscar los primeros 5 números en DATOS que tengan raíz cuadrada entera, es decir, que estén contenidos en CUAD. Además debe calcular esa raíz cuadrada y colocar los valores en ENTERO. La tabla DATOS puede tener valores repetidos.



Se debe diseñar e implementar un programa para realizar estas acciones en la tarjeta DRAGON 12+. Este programa debe incluir cuatro subrutinas:

**Subrutina LEER\_CANT:** Con esta subrutina se recibe el valor de CANT por medio del teclado, utilizando la subrutina GETCHAR. La subrutina debe validar que el valor ingresado sea un número entre 1 y 99. La subrutina LEER\_CANT debe desplegar el siguiente mensaje en pantalla:

> INGRESE EL VALOR DE CANT (ENTRE 1 Y 99):

Luego de que se presione una tecla numérica la subrutina debe validar que la tecla presionada es numérica, en caso contrario debe permanecer leyendo el teclado hasta que una tecla numérica sea presionada. Luego de leer el primer dígito la subrutina LEER\_CANT, desplegará ese valor en la pantalla y procederá a leer la segunda tecla numérica. Al recibirse la segunda tecla numérica, la subrutina debe calcular el valor ingresado como un número entero y devolver el valor en la variable CANT. El valor ingresado por el usuario debe aparecer al final del mensaje en el Terminal. Los valores entre 1 y 9 deberán ser recibidos como 01, 02..., 09. El valor 00 NO es un valor válido y no debe ser aceptado.

**Subrutina BUSCAR:** Esta subrutina busca los CANT valores de DATOS que estén en CUAD. Cada vez que encuentre un valor válido la subrutina BUSCAR debe llamar a la subrutina RAIZ. Además la subrutina debe llevar un contador de los valores encontrados en la variable CONT. La subrutina BUSCAR le pasará el valor a RAIZ por medio de la pila y por este mismo medio RAIZ devolverá el resultado. La subrutina BUSCAR debe poner este resultado en el arreglo ENTERO.

	<p style="text-align: center;"><b>UNIVERSIDAD DE COSTA RICA</b> <b>ESCUELA DE INGENIERÍA ELÉCTRICA</b></p>	
<p style="text-align: center;"><b>IE-623 MICROPROCESADORES</b> <b>Prof. Geovanny Delgado</b></p>		

**Subrutina RAIZ:** Esta subrutina calcula la raíz cuadrada . El valor al cual se le debe obtener la raíz cuadrada debe ser pasado a la subrutina por la pila y la subrutina devolverá el valor calculado por la pila.

Nota: Para calcular el valor de la raíz cuadrada se utilizará el Algoritmo Babilónico, cuya descripción se adjunta.

**Subrutina Print\_RESULT:** Esta subrutina debe imprimir los resultados de la siguiente manera:

>CANTIDAD DE VALORES A ENCONTRADOS: <CONT>

Este resultado debe aparecer dos líneas por debajo del último mensaje desplegado.

Adicionalmente esta subrutina deberá imprimir, en el terminal, el arreglo ENTERO calculado, de la siguiente manera.

> ENTERO: <num\_1> ,<num\_2> , ..., <num\_k>

Estos resultados deberán aparecer dos líneas por debajo del último mensaje desplegado.

**Nota:** Para desplegar todos los mensajes se debe utilizar la subrutina Printf.

De esta manera una corrida del programa, para:

DATOS = 4, 9, 18, 4, 27, 63, 12, 32, 36, 15.

deberá generar en el Terminal lo siguiente:

> INGRESE EL VALOR DE CANT (ENTRE 1 Y 99): 07

> CANTIDAD DE NUMEROS ENCONTRADOS : 4

> ENTERO: 2, 3, 2, 6

- a. Realice el diseño del programa que resuelve este problema.
- b. Haga el código en ensamblador para este programa. Edítelo directamente en el asmIDE. Utilice las directivas de ensamblador apropiadas para definir



**IE-623 MICROPROCESADORES**  
**Prof. Geovanny Delgado**

las localizaciones de memoria. Coloque LONG en \$1000, CANT en \$1001, CONT en \$1002, , DATOS en \$1020 y CUAD en \$1040 y ENTERO en \$1100.

- c. El programa debe colocarse en la memoria RAM a partir de la posición \$2000. El programa debe estar debidamente estructurado y documentado.
- d. Ensamble y depure el programa, hasta que genere el archivo .S19.
- e. Corra el programa en la tarjeta DRAGON 12 con un conjunto de datos de su elección y verifique que corre correctamente.
- f. Envíe su archivo .asm, a la dirección [microp@cr-automation.com](mailto:microp@cr-automation.com) a más tardar a las 9:00 a.m. de la fecha en que debe entregar su tarea. El formato del archivo debe ser SuNombre#.asm., donde # es el número de la tarea.

**ALGORITMO BABILONICO**



IE-623 MICROPROCESADORES  
Prof. Geovanny Delgado

El algoritmo babilónico se centra en el hecho de que cada lado de un cuadrado es la raíz cuadrada del área. Fue usado durante muchos años para calcular raíces cuadradas a mano debido a su gran eficacia y rapidez. Para calcular una raíz, dibuje un rectángulo cuya área sea el número al que se le busca raíz y luego aproxime la base y la altura del rectángulo hasta formar o por lo menos aproximar un cuadrado.

El algoritmo se puede enunciar sin el uso de dibujos como sigue:

CÁLCULO DE LA Raíz( $x$ ):

1 Elija dos números  $b$  y  $h$  tales que  $bh = x$

2 Si  $h \approx b$  vaya al paso 6, si no, vaya al paso 3

3 Asigne  $b \leftarrow \frac{h + b}{2}$

4 Asigne  $h \leftarrow \frac{x}{b}$

5 Vaya al paso 2

6. Escriba  $\sqrt{x} \approx b$

Este algoritmo aproxima la raíz cuadrada de cualquier número real tanto como se desee. Es claro que no se necesita conocer el valor de  $h$ , puesto que depende directamente de  $x$  y que el área del rectángulo siempre se aproxima a la raíz cuadrada de  $x$  sin importar el valor de  $b$  siempre y cuando  $b > 0$ .

De esta manera surge la función recursiva

$$f_0(x) = x$$
$$f_n(x) = \frac{1}{2} \left( \frac{x}{f_{n-1}(x)} + f_{n-1}(x) \right)$$



IE-623 MICROPROCESADORES  
Prof. Geovanny Delgado

de manera tal que  $n$  es la  $n$ -ésima aproximación a  $\sqrt{x}$ . Esto implica que

$$f_{\infty}(x) = \sqrt{x}$$

Puesto que algunas raíces son números irracionales es necesario definir qué tanto es "aproximadamente". Afortunadamente nadie es capaz de escribir un número con una infinita cantidad de dígitos, por lo que el umbral de aproximación se limita a la cantidad de dígitos que se es capaz de escribir. Entonces podemos definir que el algoritmo termine en el momento que la última aproximación es la misma que la anterior (es decir, ya no se puede aproximar más).

### DESCRIPCION FORMAL DEL ALGORITMO.

De manera formal, se expresa el algoritmo babilónico usando pseudocódigo de la siguiente manera:

**Función** raíz( $x$ )

$r \leftarrow x$   
 $t \leftarrow 0$

**mientras**  $t \neq r$

$t \leftarrow r$

$$r \leftarrow \frac{1}{2} \left( \frac{x}{r} + r \right)$$

**devolver**  $r$