Práctica No. 1.

Calculadora básica de 8 bits usando lenguaje ensamblador del microcontrolador MC9S08JM60.

Dinoingenieros
250 millones de años
a su servicio

Ingenieros Diseñadores:

Yorguin José Mantilla Ramos 1127617499 yorguinj.mantilla@udea.edu.co	Manuela Restrepo Cardona 1152704892 manuela.restrepoc@udea.edu.co

a. Descripción del sistema

El sistema consiste en una calculadora simple hecha con el microcontrolador MC9S08JM60. La calculadora cuenta con cuatro operaciones básicas: suma, resta, multiplicación y división entera. Solo se manejan números enteros con signo representables en 8 bits (-128 a 127). La visualización del resultado se realiza en memoria mediante código ASCII.

b. Descripción de requisitos funcionales

- El usuario ingresa los números que desea operar mediante 8 switches del **puerto B** de la tarjeta, utilizando el formato de complemento a 2 de 8 bits.
- Luego, el usuario debe seleccionar el operando que ingresa (A o B) mediante los botones de captura correspondientes (**Capture A** y **Capture B**), ubicados en los 4 bits inferiores del **puerto A** de la tarjeta. Para referencia ver el **esquema de conexiones**.
- Similarmente, el usuario ingresa la operación que desea realizar mediante dos switches asociados a los 2 bits inferiores del **puerto C** de la tarjeta. El código binario para las operaciones disponibles es el siguiente:

00	Suma
01	Resta
10	Multiplicación
11	División Entera

- Y captura su selección de la operación mediante el botón **Capture Op**
- Finalmente, debe presionar el botón de **start** para ejecutar la operación y poder visualizar el resultado en la dirección **0x130** de la memoria.
- En caso de que se haya identificado un error durante el cálculo (ya sea por overflow o por operaciones indebidas como la división por 0), se tiene un indicador de error en la memoria para el usuario (para referencia ver la **especificación de interfaces**).

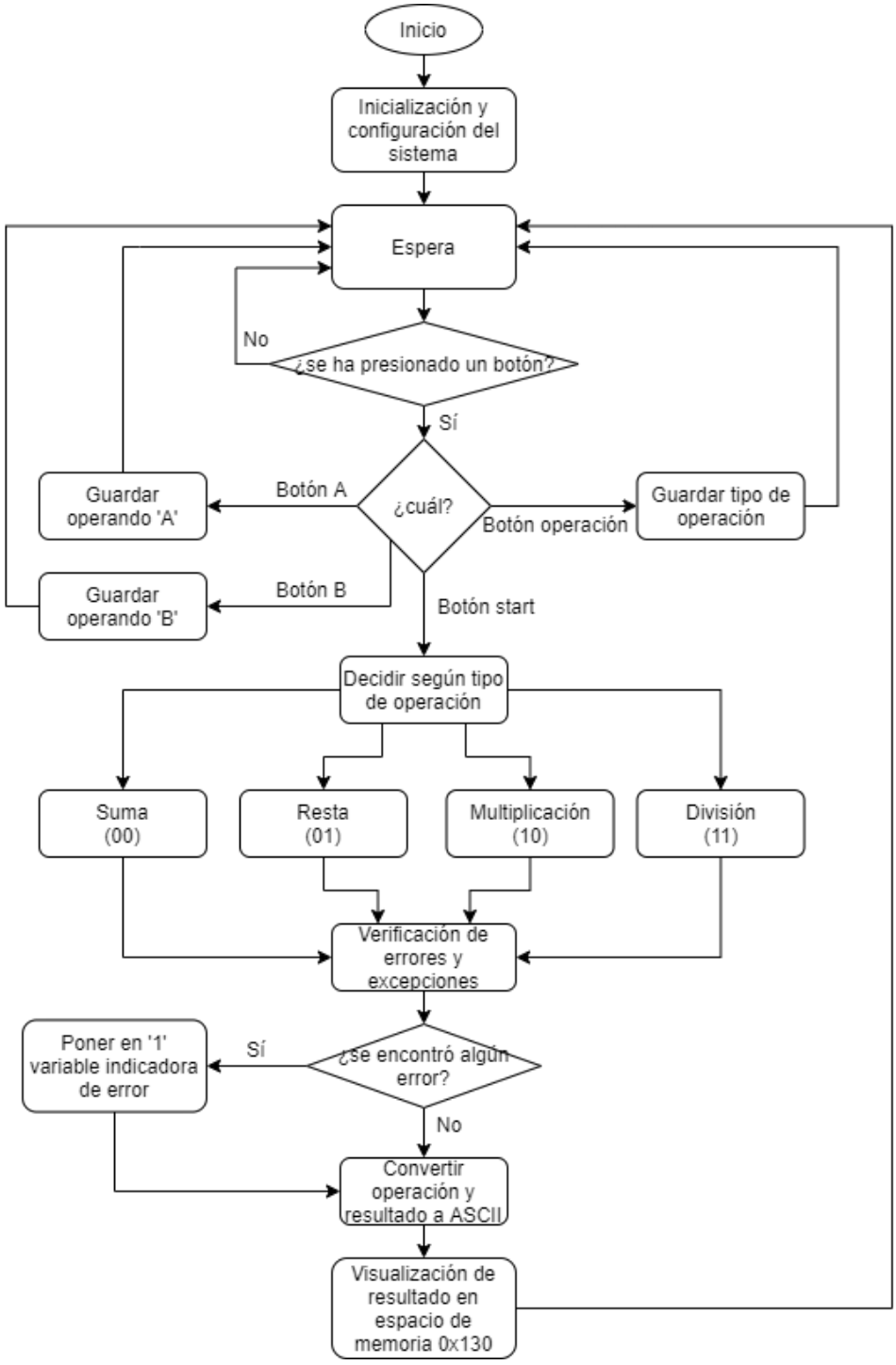
c. Enumeración de requisitos no funcionales

La calculadora NO cuenta con las siguientes funciones:

- Entrada de datos mediante sistema decimal
- Display de visualización
- No opera números fraccionarios.
- No muestra el residuo de la división entera

d. Arquitectura básica del sistema

Desde el punto de vista funcional el sistema se puede diagramar de la siguiente manera:



e. Especificación de interfaces

Entrada

No hay interfaces, se ingresan los operandos y la operación mediante switches y botones como es mostrado en el **esquema de conexiones** y explicado en la **descripción de requisitos funcionales**.

Salida

Los resultados se visualizan en memoria, específicamente desde la dirección **0x0130** hasta la dirección **0x13E**. En ese espacio se mostrará la operación indicada por el usuario y su resultado mediante símbolos en ASCII. En caso de error la posición **0x13E** tendrá una **E**, y en tal caso el resultado mostrado se debe ignorar por el usuario. Si el resultado es correcto la casilla **0x13E** tendrá un carácter de espacio en blanco.

Para ilustrar lo anterior se tienen 2 ejemplos, uno con un resultado correcto y otro con un resultado incorrecto:

- Al ejecutar 13 multiplicado por 3, obtendremos el siguiente mensaje en memoria:

Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
0130	+	0	1	3	x	+	0	0	3	=	+	0	3	9	

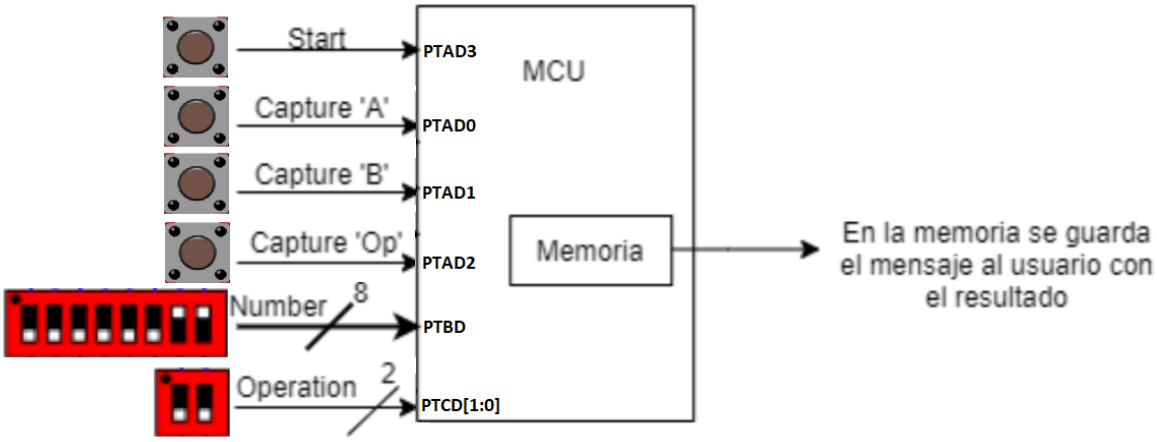
Notemos que la casilla E esta con un espacio en blanco al no haber error.

- Al ejecutar 127 multiplicado por 127, obtendremos el siguiente mensaje en memoria:

Address	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E
0130	+	1	2	7	x	+	1	2	7	=	+	0	0	1	E

Notemos que la casilla E muestra la letra **E**, para indicar que hubo un error durante la operación; en este caso ocasionado debido a la imposibilidad de representar 16129 en 8 bits con signo. En correspondencia, el resultado “+001” debe ignorarse o tomarse como incorrecto por el usuario.

f. Esquema de conexiones



g. Limitaciones y posibles mejoras

- Solo acepta números en el rango de -128 a 127 tanto en la entrada como en la salida. Esto podría aumentarse pero haría más compleja la implementación de las operaciones ya qué la ALU soporta operaciones en su mayoría de 8 bits.
- Se implementa la división entera, sin embargo la calculadora en sí no muestra el residuo al usuario. Esto podría implementarse con una rutina especial para este caso donde la operación retorna no un solo número sino dos.
- La calculadora funciona solo para números enteros con signo. Permitir el ingreso de números decimales implicaría la implementación de rutinas para el manejo de números en punto flotante y evidentemente un ingreso de datos mediante números decimales con un teclado numérico (ya qué pedir el ingreso de números en punto flotante mediante switches implicaría demasiada complejidad para el usuario).
- El resultado de la multiplicación se puede retornar internamente hasta 16 bits (unsigned) sin problema; la limitante de restringirlo a un número de 8 bits surge más en las rutinas de conversión a ASCII donde se deben realizar divisiones sucesivas por potencias de 10 y donde el residuo se guarda en un registro de 8 bits para determinar el siguiente dígito. Lo anterior implica que como máximo se puede aceptar un residuo de 255 (unsigned), lo cuál no podemos garantizar qué siempre ocurra. Por esta razón se optó por restringir el espacio de resultados mostrados al usuario a números de 8 bits; garantizando así un correcto control de nuestros resultados (tanto correctos como incorrectos). Todo esto se podría mejorar implementando nuestra propia rutina de división qué acepte residuos de 16 bits.