

**Programa:**

Programa con operaciones aritméticas

**Materia**:

Lenguajes de interfaz

**Docente**:

Ing. Claudia Elena Carrillo Reyes

**Equipo 7:**

|  |  |
| --- | --- |
| Nº Control | Alumno |
| 21130599 | Gerardo Enrique Ramos Espinoza |
| 22130584 | José Ramón Romero Zavala |

**Lugar:** Torreón, Coahuila. **Fecha**: 26/Noviembre/2024

# Índice.

[**Indice. 1**](#__RefHeading___Toc1367_1871765703)

[**Enunciado. 2**](#__RefHeading___Toc459_2245012153)

[**Resumen del programa. 2**](#__RefHeading___Toc1371_1871765703)

[**Descripción del programa. 3**](#__RefHeading___Toc1373_1871765703)

[**Código fuente del programa 8**](#__RefHeading___Toc461_2245012153)

[**Capturas de la ejecución del programa 16**](#__RefHeading___Toc458_1391150105)

[**Bibliografía 18**](#__RefHeading___Toc1375_1871765703)

# Enunciado.

Elaborar un programa que realice dos aplicaciones (tareas o funciones) utilizando en su código, varias instrucciones del grupo de aritméticas.

El programa tiene como requisitos para su presentación:

1. Funcionando al 100%.
2. Pantalla con el número de equipo, los nombres de los alumnos y nombre del programa (asignado por el equipo).
3. Mensaje(s) para darle indicaciones al usuario y para el (los) resultado(s) del programa.

# Resumen del programa.

El programa, escrito en lenguaje ensamblador para el modelo **x86 (32 bits)**, permite realizar operaciones las operaciones de división y resta, entre dos números de un dígito ingresados por el usuario. Incluye funcionalidades para manejar errores, como división entre cero y desbordamiento aritmético, así como validación de entrada, permitiendo únicamente introducir caracteres numéricos y en el menú únicamente las opciones disponibles. El programa guía al usuario a través de un menú interactivo y muestra los resultados de manera clara, junto con mensajes de error en caso de entrada no válida. Su propósito es ofrecer un menú principal con tres opciones:

1. **Mostrar Créditos**: Presenta información sobre los creadores del programa, incluyendo nombres y matrículas.
2. **Ejecutar Calculadora**: Permite realizar operaciones de resta y división entre números enteros, manejando casos especiales como errores por división entre cero, desbordamiento y cuando el primer número es menor que el segundo.
3. **Salir**: Finaliza la ejecución del programa.

El programa incluye funciones para gestionar entrada y salida de datos en pantalla, utiliza interrupciones del sistema para interactuar con el hardware y valida las entradas del usuario para garantizar operaciones correctas. Está estructurado en secciones claras que cubren la funcionalidad del menú, las operaciones aritméticas y los mensajes de error. Además, emplea una interfaz básica de texto que limpia y actualiza la pantalla dinámicamente según las acciones del usuario.

# Descripción del programa.

**Tabla de registros utilizados en el programa.**

|  |  |  |
| --- | --- | --- |
| **Registros** | **Uso Principal** | **Descripción** |
| AX | Segmento de datos, acumulador para cálculos. | Configura el segmento de datos (mov ax, @data), realiza cálculos como suma, resta y división. |
| **BX** | Almacenamiento del multiplicador/divisor en operaciones aritméticas. | Se utiliza en operaciones como mul bl y div bl. |
| **CX** | No utilizado explícitamente en este programa. | Disponible para bucles o contadores en posibles extensiones. |
| DX | Dirección de cadenas para mostrar en pantalla. | Utilizado con int 21h (función 09h) para referenciar las cadenas de texto. |
| **AL** | Almacena valores de entrada y resultados intermedios. | Maneja valores numéricos en operaciones aritméticas (add, sub, mul, div). |
| **BL** | Almacena el operador o el segundo número para cálculos. | Contiene el valor del multiplicador/divisor en operaciones como mul y div. |
| **AH** | Código de función para interrupciones DOS. | Define el tipo de operación a realizar con int 21h (e.j., mostrar texto, leer caracteres). |
| **SP** | No utilizado directamente. | Maneja la pila en operaciones internas de interrupciones. |
| **DS** | Segmento de datos activo. | Configurado al inicio (mov ds, ax) para acceder a variables en el segmento .DATA. |
| FLAGS | Indicador de condiciones (e.g., desbordamiento). | Detecta desbordamiento aritmético con el flag de Overflow (OF), evaluado mediante jo. |

Los registros de propósito general del microprocesador x86 se dividen en segmentos específicos de 16 y 8 bits, permitiendo manejar datos y direcciones de manera eficiente en el entorno del modo real o virtual. En el programa analizado, se utilizan los registros AX, BX, CX, DX, y sus subdivisiones (AL, AH, BL, etc.), los cuales cumplen funciones críticas en operaciones aritméticas, control de flujo, y manipulación de datos.

**Descripción General de los Registros Utilizados**

1. **AX (Acumulador):**

- Este registro es ampliamente utilizado para operaciones aritméticas y de manipulación de datos. En este programa, su parte inferior (**AL**) es utilizada para almacenar valores de entrada, realizar operaciones de suma, resta y división, y para almacenar resultados intermedios.

- AX puede dividirse en dos registros de 8 bits: **AH** (parte alta) y **AL** (parte baja). Esto permite manejar datos de menor tamaño sin afectar el registro completo.

2. **BX (Base):**

- Usado como un registro base para operaciones como multiplicación y división. Su parte baja (**BL**) contiene el segundo operando en cálculos aritméticos, como el divisor en divisiones y el multiplicador en multiplicaciones.

- También puede dividirse en **BH** y **BL** para operaciones de precisión más fina.

3. **CX (Contador):**

- Aunque no se utiliza explícitamente en este programa, es conocido por ser el registro de contador por defecto en instrucciones de bucles y operaciones repetitivas, como `LOOP` o desplazamientos.

4. **DX (Datos):**

- Utilizado principalmente para manejar direcciones de cadenas en operaciones de entrada/salida. En el programa, DX almacena las direcciones de las cadenas que se muestran en pantalla mediante la interrupción **int 21h**.

- Al igual que otros registros, DX puede dividirse en **DH** y **DL**.

5. **SP (Puntero de Pila):**

- Aunque no se utiliza directamente en este programa, SP gestiona la pila interna del sistema, particularmente durante el manejo de interrupciones.

6. **DS (Segmento de Datos):**

- Es el registro que define el segmento de datos activo. En este programa, DS se configura al inicio (`mov ds, ax`) para acceder a las variables declaradas en el segmento `.DATA`.

7. **FLAGS (Indicadores):**

- Este registro no se utiliza directamente por el programador, pero su indicador de **Overflow (OF)** es crítico para detectar errores de desbordamiento en operaciones aritméticas. El programa lo evalúa mediante la instrucción `jo` (jump if overflow).

**Uso en Direccionamiento**

Los registros de propósito general pueden participar en el direccionamiento indirecto, permitiendo referenciar posiciones de memoria mediante combinaciones de base, índice y desplazamiento. En este programa, se usan principalmente para manipular datos en memoria y realizar cálculos.

**Ejemplo de Direccionamiento:**

- **MOV AL, [BX]:** Accede al contenido de la memoria apuntada por BX.

- **ADD AL, [DX]:** Suma el valor en la posición de memoria referenciada por DX con el contenido de AL.

En el modo real, los registros como DS son usados por defecto como segmento de datos. Sin embargo, el manejo explícito del segmento puede variar en modos protegidos o avanzados.

Consideraciones en Modo Real

En el modo real, los registros de 16 bits son fundamentales para garantizar la compatibilidad con sistemas más antiguos. El acceso a combinaciones como **BX + SI** o **BP + DI** es común, pero siempre respetando los límites de 64 KB del segmento. El programa evita referencias indirectas complejas, priorizando la simplicidad en las operaciones de entrada/salida y cálculos aritméticos.

1. **Inicio y Configuración:**
   * El programa comienza configurando los segmentos de datos y el segmento de pila para inicializar el entorno de ejecución.
   * Se limpia la pantalla usando una llamada al servicio de video int 10h para presentar el menú principal de forma ordenada.
2. **Menú Principal:**
   * Se muestra un menú interactivo con las opciones:
     1. **Mostrar Créditos:** Presenta la información del equipo desarrollador, el nombre del programa y los integrantes.
     2. **Ejecutar Calculadora:** Permite realizar operaciones de resta y división entre dos números.
     3. **Salir:** Finaliza la ejecución del programa.
   * El usuario elige una opción introduciendo el número correspondiente.
3. **Mostrar Créditos:**
   * Al seleccionar la opción 1, se limpia nuevamente la pantalla y se despliega información sobre el equipo, incluyendo los nombres de los integrantes, el número del equipo, y el título del programa.
   * Después de visualizar los créditos, el usuario puede regresar al menú principal presionando ENTER.
4. **Calculadora de Resta y División:**
   * **Entrada de Datos:**
     1. Se solicita al usuario que introduzca dos números (entre 0 y 999) y el operador deseado (- para resta o / para división).
     2. Los números se ingresan carácter por carácter y se validan para asegurar que estén dentro del rango permitido.
   * **Operaciones Matemáticas:**
     1. Si el operador es -, el programa realiza una resta, verificando que el primer número sea mayor o igual al segundo.
     2. Si el operador es /, se realiza la división, asegurándose de que el divisor no sea cero.
   * **Manejo de Errores:**
     1. Si ocurre un error, como división por cero, un número menor al otro en una resta, o desbordamiento aritmético, se muestra un mensaje de error específico.
   * **Resultados:**
     1. El resultado de la operación se calcula y se presenta en la pantalla. Si se requiere convertir un valor numérico a ASCII, se realiza con un bucle que convierte cada dígito.
5. **Repetición:**
   * Después de completar una operación o visualizar un error, se pregunta al usuario si desea volver a realizar otra operación. Si elige "S" o "s", la calculadora reinicia; de lo contrario, regresa al menú principal.
6. **Salida del Programa:**
   * Si el usuario selecciona la opción de salir (opción 3), el programa finaliza limpiando la pantalla y liberando los recursos.

**Técnicas y Servicios Utilizados:**

* **Interrupciones del BIOS y DOS:**
  + int 10h: Control de pantalla para posicionar texto y limpiar el área visible.
  + int 21h: Entrada/salida de caracteres y cadenas, con funciones como lectura de teclas (ah=01h) y visualización de texto (ah=09h).
* **Control de Flujo:**
  + El programa utiliza etiquetas y saltos (jmp) para navegar entre secciones, siguiendo un diseño estructurado, pero sin subrutinas.
* **Validación de Entrada:**
  + Las entradas del usuario son validadas para garantizar que cumplen con los requisitos de rango y formato.

Una de las instrucciones no utilizadas previamente en clase es la de TEST. La instrucción TEST en ensamblador x86 es utilizada para realizar una operación lógica **AND** entre dos operandos, pero sin almacenar el resultado. Su propósito principal es actualizar las banderas **de estado** del registro **EFLAGS**, lo que permite realizar decisiones basadas en el resultado de la operación. Funciona de la siguiente manera:

1. Realiza una operación lógica **AND** bit a bit entre los operandos.
2. **No modifica** el contenido de los operandos.
3. Actualiza las **banderas del registro EFLAGS**:
   * **ZF (Zero Flag):** Se activa si el resultado de la operación AND es 0.
   * **SF (Sign Flag):** Indica si el resultado tiene el bit más significativo (MSB) activado (negativo en representación con signo).
   * **PF (Parity Flag):** Indica si el número de bits con valor 1 en el resultado es par.
   * **OF (Overflow Flag):** Siempre se desactiva con TEST porque no hay desbordamiento.
   * **CF (Carry Flag):** Siempre se desactiva con TEST porque no hay acarreo.

En todo el programa utilizamos saltos para controlar el flujo del programa, estos permiten alterar el flujo del programa basado en las banderas del registro **EFLAGS**. Estas banderas reflejan el resultado de la última operación ejecutada, como comparaciones o pruebas lógicas. Los saltos condicionales verifican estas banderas para decidir si continuar la ejecución secuencial o dirigirse a otra parte del programa. Los siguientes son los saltos utilizados:

**1. JE (Jump if Equal)**

* **Descripción**: Salta si los operandos comparados son iguales.
* **Bandera evaluada**:
  + **ZF (Zero Flag)**: Salta si ZF = 1 (el resultado de la comparación fue igual a 0).

**2. JB (Jump if Below)**

* **Descripción**: Salta si el primer operando es menor que el segundo en comparación **sin signo**.
* **Bandera evaluada**:
  + **CF (Carry Flag)**: Salta si CF = 1 (indica que hubo acarreo en una comparación sin signo).

**3. JA (Jump if Above)**

* **Descripción**: Salta si el primer operando es mayor que el segundo en comparación **sin signo**.
* **Banderas evaluadas**:
  + **CF (Carry Flag)** y **ZF (Zero Flag)**:
    - Salta si CF = 0 y ZF = 0 (sin acarreo y los operandos no son iguales).

**4. JO (Jump if Overflow)**

* **Descripción**: Salta si ocurrió un desbordamiento en la última operación aritmética.
* **Bandera evaluada**:
  + **OF (Overflow Flag)**: Salta si OF = 1 (hubo desbordamiento).

**5. JNZ (Jump if Not Zero)**

* **Descripción**: Salta si el resultado de la última operación no fue 0.
* **Bandera evaluada**:
  + **ZF (Zero Flag)**: Salta si ZF = 0 (el resultado no fue igual a 0).

Estas instrucciones permiten implementar lógica condicional en ensamblador, como bifurcaciones y bucles, según los resultados de operaciones anteriores.

También se usan instrucciones para almacenar y acceder a datos de la pila. Las instrucciones PUSH y POP son utilizadas para interactuar con la **pila**, una estructura de datos en memoria que sigue el principio **LIFO** (Ultimo en entrar, Primero en salir).

1. **PUSH**: Guarda un valor en la parte superior de la pila y ajusta el puntero de la pila (**SP**) hacia abajo.
2. **POP**: Recupera el último valor almacenado en la pila (el que está en la parte superior) y ajusta el puntero de la pila hacia arriba.

La pila es crucial para almacenar temporalmente datos como registros, direcciones de retorno, o valores intermedios durante cálculos.

# Código fuente del programa

# Capturas de la ejecución del programa

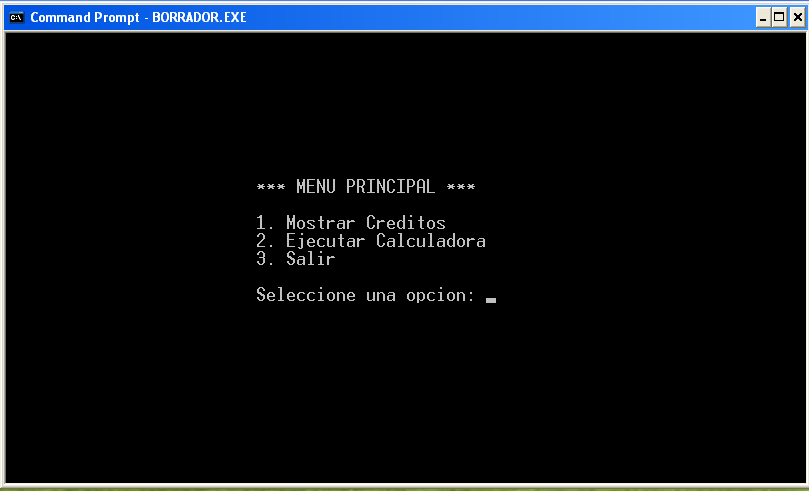


Ilustración Menú principal.

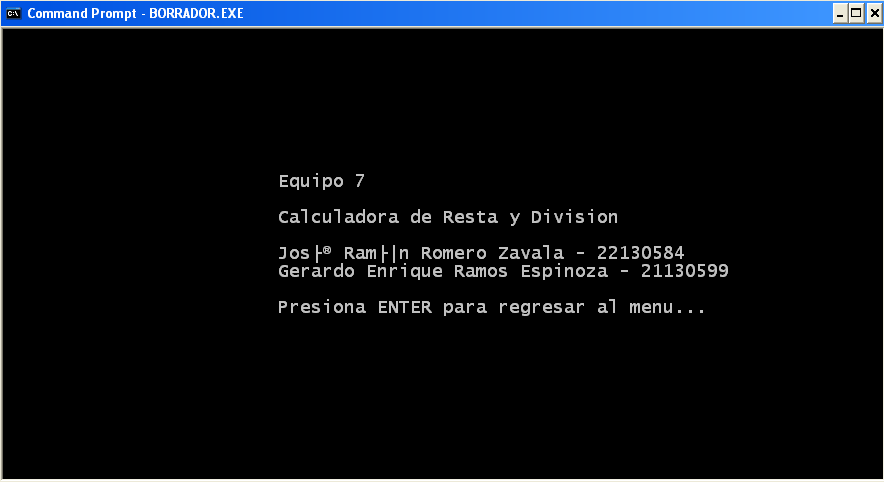


Ilustración Opción 1, créditos del programa.

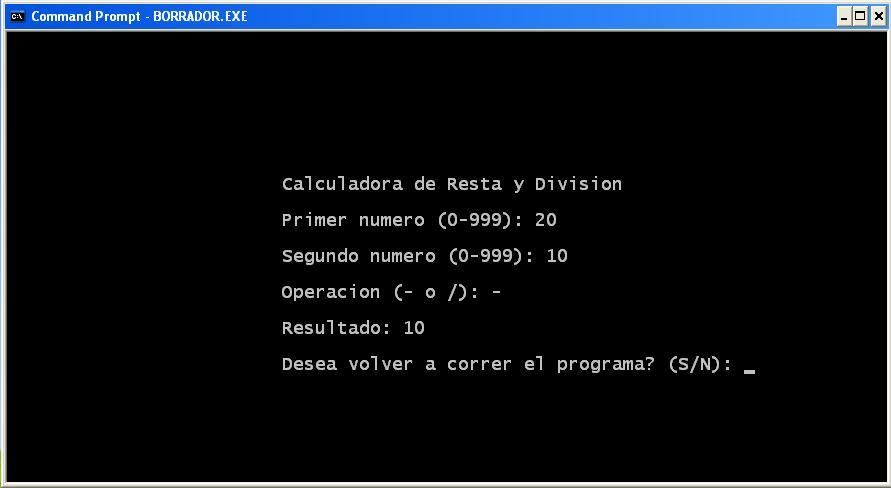


Ilustración Opción 2, calculadora, realizando una resta.

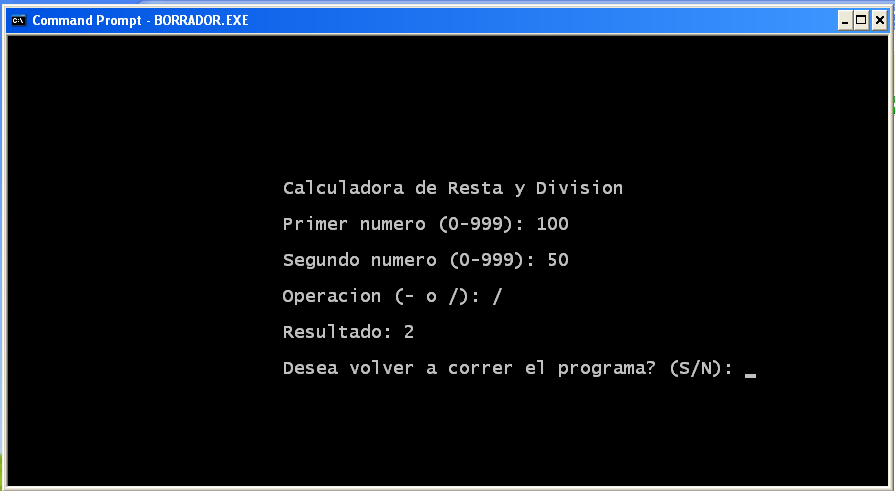


Ilustración Opción 2, calculadora, realizando una división.

# Bibliografía

* Corporation, I. (1986). *80386 Programmer’s Reference Manual*.
* CRN. (s. f.). <http://web.archive.org/web/20071012212412/https://www.crn.com/crn/special/supplement/816/816p65_hof.jhtml>
* Christopher L. Morgan & Mitchell Waite, Introducción al Microprocesador 8086/8088 (16 bits), Editorial Mc.Graw-Hill.
* William H. Murray III && Chris H Pappas, 80386/80286 Programación en Lenguaje Ensamblador, Editorial Mc Graw Hill.