Seminario de Solución de problemas de Traductores de Lenguajes I

Centro Universitario de Ciencias Exactas en ingenierías

Universidad de Guadalajara

Maestro: Tonatiuh Hernandez Casas

Juan Antonio Pérez Juárez
Código: 215660996
Carrera: INCO

Actividad Integradora

Marco Teórico

Primero debemos definir lo que es el EMU8086.
El emu8086 es un emulador del microprocesador 8086 (Intel o AMD compatible) con assembler integrado. A diferencia del entorno de programación en assembler utilizado anteriormente en la cátedra (MASM), este entorno corre sobre Windows y cuenta con una interfaz gráfica muy amigable e intuitiva que facilita el aprendizaje el leguaje de programación en assembler.

Dado que en un entorno emulado de microprocesador no es posible implementar una interfaz real de entrada/salida, el emu8086 permite interfacear con dispositivos virtuales y emular una comunicación con el espacio de E/S. Para esto, el emu8086 cuenta con una serie de dispositivos virtuales preexistentes en el software base, listos para ser utilizados, entre los que se encuentran una impresora, un cruce de calles con semáforos, un termómetro, un motor paso a paso, etc. No obstante, la cátedra ha desarrollado dispositivos adicionales con características particulares para la realización del segundo trabajo práctico.

Así que se hará el desarrollo de una calculadora simple en este entorno desarrollo haciendo uso de el lenguaje ensamblador, veamos un poco de lo que es ensamblador.

El lenguaje ensamblador o assembler (en inglés: assembler language y la abreviación asm) es un lenguaje de programación que se usa en los microprocesadores. Implementa una representación simbólica de los códigos de máquina binarios y otras constantes necesarias para programar una arquitectura de procesador y constituye la representación más directa del código máquina específico para cada arquitectura legible por un programador. Cada arquitectura de procesador tiene su propio lenguaje ensamblador que usualmente es definida por el fabricante de hardware, y está basada en los mnemónicos que simbolizan los pasos de procesamiento (las instrucciones), los registros del procesador, las posiciones de memoria y otras características del lenguaje. Un lenguaje ensamblador es por lo tanto específico de cierta arquitectura de computador física (o virtual). Esto está en contraste con la mayoría de los lenguajes de programación de alto nivel, que idealmente son portables.

Un programa utilitario llamado ensamblador es usado para traducir sentencias del lenguaje ensamblador al código de máquina del computador objetivo. El ensamblador realiza una traducción más o menos isomorfa (un mapeo de uno a uno) desde las sentencias mnemónicas a las instrucciones y datos de máquina. Esto está en contraste con los lenguajes de alto nivel, en los cuales una sola declaración generalmente da lugar a muchas instrucciones de máquina.

Veamos algunas de sus ventajas:

- El código escrito en lenguaje ensamblador posee una cierta dificultad de ser entendido ya que su estructura se acerca al lenguaje máquina, es decir, es un lenguaje de bajo nivel.
- El lenguaje ensamblador es difícilmente portable, es decir, un código escrito para un microprocesador, puede necesitar ser modificado, para poder ser usado en otra máquina distinta. Al cambiar a una máquina con arquitectura diferente, generalmente es necesario reescribirlo completamente.

Pero ahora pasemos al desarrollo de esta actividad.

Desarrollo de la Actividad

Desarrollar una calculadora en lenguaje ensamblador implementando las operaciones aritméticas básicas (Suma, resta, multiplicación, división y potencia).

Así que ahora veamos el código en Ensamblador para hacer esto posible.

```
Unset
;Hecho por Juan Antonio Perez Juarez
;INCO 2024B
;215660996

.model small
.stack 100h
.data
    ;Inicio del segmento de datos
    msg1 db 10,13,'Ingrese el primer numero: $'
    msg2 db 10,13,'Ingrese el segundo numero: $'
    msgSuma db 10,13,'Suma: $'
    msgResta db 10,13,'Resta: $'
    msgMult db 10,13,'Multiplicacion: $'
    msgDiv db 10,13,'Division: $'
```

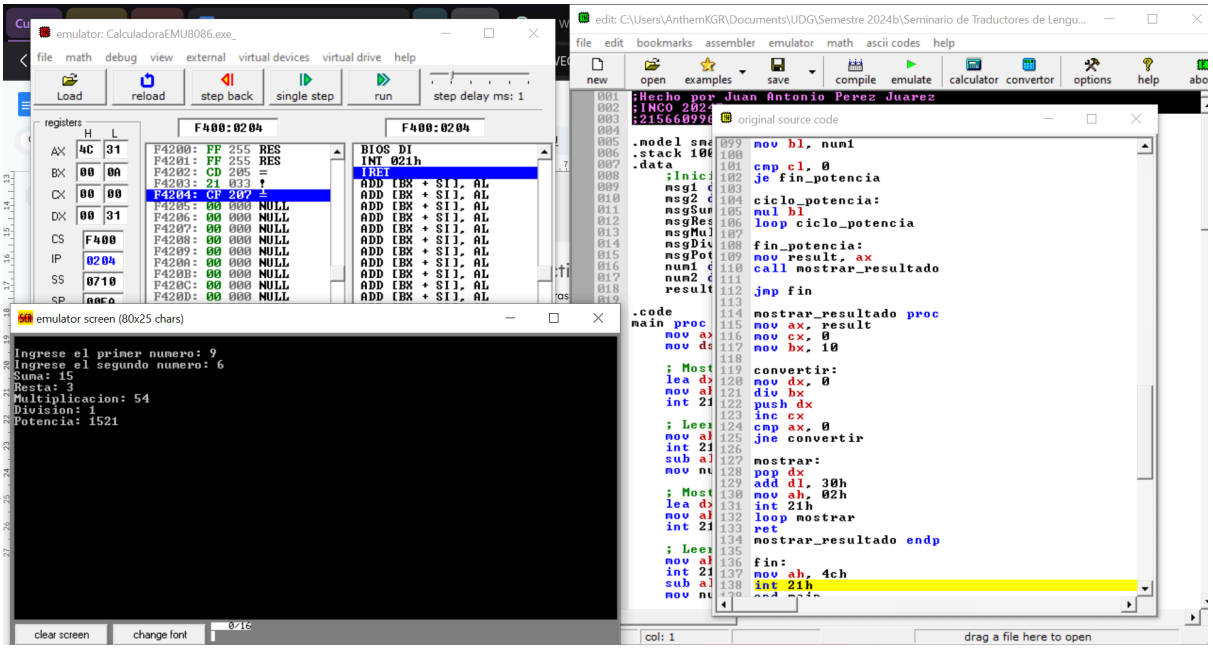
```
    msgPot db 10,13,'Potencia: $'
    num1 db ?
    num2 db ?
    result dw ?

.code
main proc
    mov ax, @data
    mov ds, ax

    ; Mostrar mensaje para primer numero
    lea dx, msg1
    mov ah, 09h
    int 21h

    ; Leer primer numero
    mov ah, 01h
    int 21h
    sub al, 30h
    mov num1, al

    ; Mostrar mensaje para segundo numero
    lea dx, msg2
    mov ah, 09h
    int 21h

    ; Leer segundo numero
    mov ah, 01h
    int 21h
    sub al, 30h
    mov num2, al

    ; SUMA
    lea dx, msgSuma
    mov ah, 09h
    int 21h

    mov al, num1
    add al, num2
    mov ah, 0
    mov result, ax
    call mostrar_resultado

    ; RESTA
    lea dx, msgResta
    mov ah, 09h
    int 21h
```

```
    mov al, num1
    sub al, num2
    mov ah, 0
    mov result, ax
    call mostrar_resultado

    ; MULTIPLICACIÓN
    lea dx, msgMult
    mov ah, 09h
    int 21h

    mov al, num1
    mul num2
    mov result, ax
    call mostrar_resultado

    ; DIVISIÓN
    lea dx, msgDiv
    mov ah, 09h
    int 21h

    mov ax, 0
    mov al, num1
    div num2
    mov ah, 0
    mov result, ax
    call mostrar_resultado

    ; POTENCIA
    lea dx, msgPot
    mov ah, 09h
    int 21h

    mov cx, 0
    mov cl, num2
    mov ax, 1
    mov bl, num1

    cmp cl, 0
    je fin_potencia

ciclo_potencia:
    mul bl
    loop ciclo_potencia

fin_potencia:
    mov result, ax
    call mostrar_resultado
```

```
    jmp fin

mostrar_resultado proc
    mov ax, result
    mov cx, 0
    mov bx, 10

convertir:
    mov dx, 0
    div bx
    push dx
    inc cx
    cmp ax, 0
    jne convertir

mostrar:
    pop dx
    add dl, 30h
    mov ah, 02h
    int 21h
    loop mostrar
    ret
mostrar_resultado endp

fin:
    mov ah, 4ch
    int 21h
end main
```

Resultados de la Actividad

Ahora veamos unas cuantas capturas de pantalla de lo que hace este código.



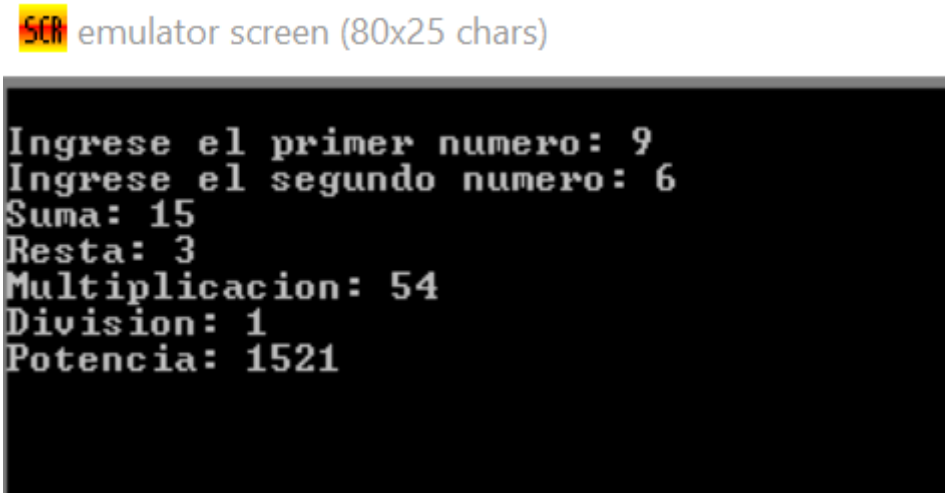
Así se ve una ejecución del código completa usando 2 numeros

El flujo de desarrollo del programa funciona de esta manera.

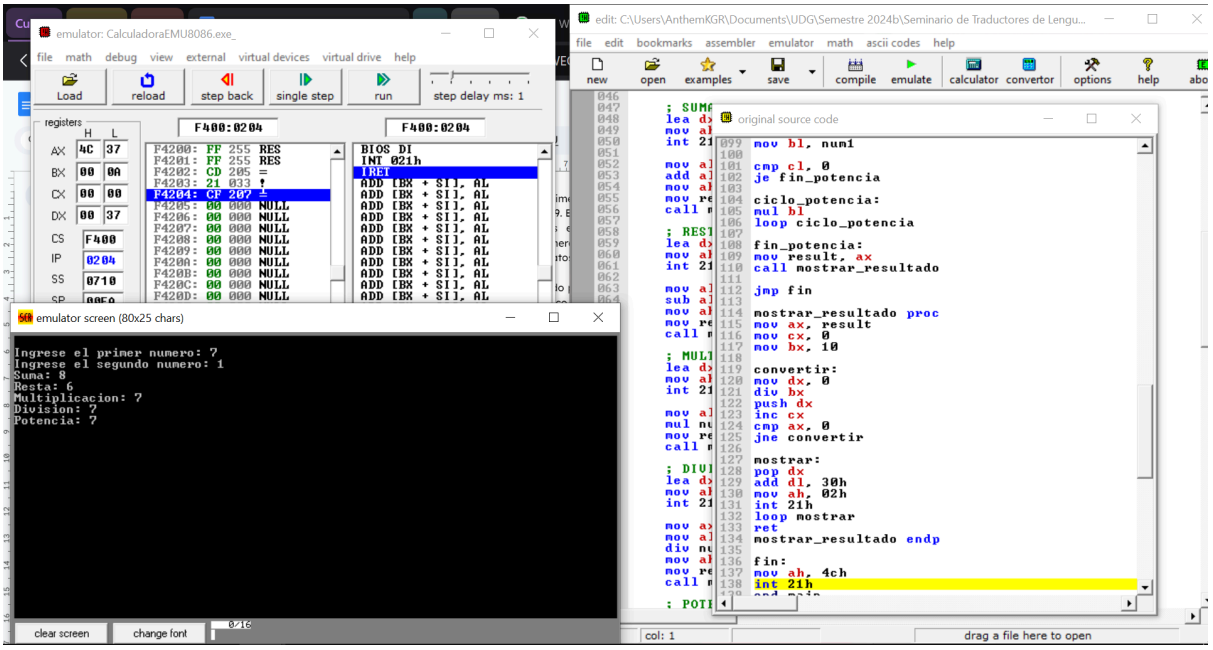
```
[INICIO]
   |
   ├─→ Inicialización de segmentos
   |
   ├─→ Entrada primer número
   |
   ├─→ Entrada segundo número
   |
   ├─→ Operaciones
   |     ├─→ Suma
   |     ├─→ Resta
   |     ├─→ Multiplicación
   |     ├─→ División
   |     └─→ Potencia
   |
   ├─→ Mostrar resultados
   |     └─→ Conversión y display
   |
   └─→ [FIN]
```

Sigue el flujo de este diagrama que hice en asciiflow.
Primero Inicia los segmentos, eso es sencillo.

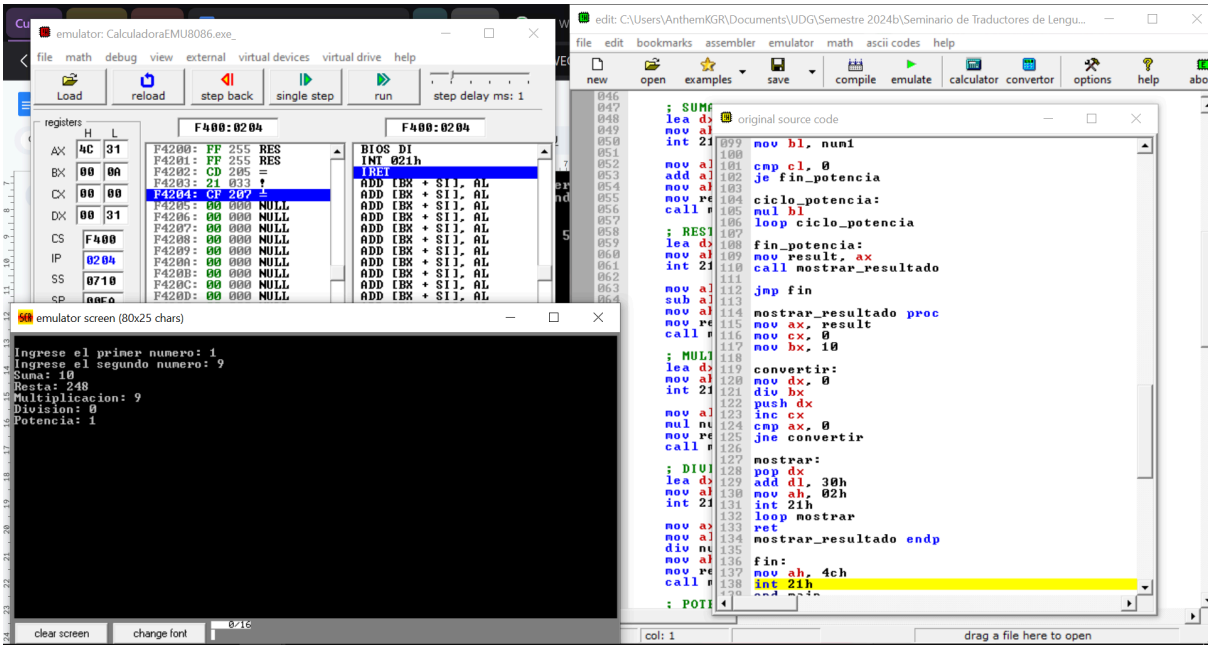
Y pide al usuario que escriba el primer número, por cuestiones de tiempo solo pude hacer que fueran números del 0 al 9. En ambos casos, en los dos números.
Una vez guardado los números en una dirección de memoria comienza a procesarlos para hacerlos de manera automática, sin que el usuario haga nada, cuando termina de procesar los datos. Los despliega en la consola para comprobar su resultado.
Aunque no está del todo terminado porqué no puede manejar la división por cero, en el apartado de división y tampoco maneja los números negativos en la resta.



Es por eso que trato de poner el numero mas grande al inicio por que sigue la lógica de programación de Numero1 menos Numero2.
Así me evito quedar en ridículo porque sale un número negativo sin convertir a Ascii. (Que trucazo, ¿No?)



Otra ejecución



Y pues ya una con un número negativo, ya que, ni modo.

Reflexión

Realmente me odie mucho en esta actividad, procrastiné demasiado lo que causó que no pudiera hacer una parte fundamental del proyecto. La graficación del seno y coseno, pero si hubiera tenido al menos otros dos días creo que hubiera podido hacerlo, además de que fue una semana muy pesada en el trabajo, no es justificación pero fue un factor que me mermó, en fin, dejando de lado las excusas, para mí este programa fue realmente desafiante, como le había comentado en reportes anteriores no soy bueno en lógica de programación y menos en un lenguaje de programación tan complicado como lo es ensamblador.
Además de tener todas las tentaciones de hacerlo con IA, como mi buen amigo ChatGPT, pero creo que es algo digno de presentar, no es mucho, pero es mi trabajo honesto, el viernes fue cuando ya no pude avanzar en el código y la verdad use tutoriales y páginas de dudosa procedencia pero todo bien.

Bibliografía:

colaboradores de Wikipedia. (2024, November 13). Lenguaje ensamblador. Wikipedia, La Enciclopedia Libre. https://es.wikipedia.org/wiki/Lenguaje_ensamblador

De Arquitectura, V. T. L. E. (2016, November 16). Emulador 8086. Arquitectura De Hardware. https://arquitectura9728.wordpress.com/2016/11/16/emulador-8086/

ASCIIFlow. (n.d.). https://asciiflow.com/#/

Ensamblador - CALCULADORA EN ASSEMBLER. (n.d.). https://www.lawebdelprogramador.com/foros/Ensamblador/1292366-CALCULADORA-EN-ASSEMBLER.html

ojoanalogo/simple-asm-calc: 🧮 Cálculadora simple hecha en asm 8086. (n.d.). GitHub. https://github.com/ojoanalogo/simple-asm-calc

About Calculator in asm 8086 - 2 Questions. (n.d.). Stack Overflow. https://stackoverflow.com/questions/36811810/about-calculator-in-asm-8086-2-questions

Simple-Calculator-Using-Assembly-Language/Simple Calculator.asm at master · IbrahiimKhalil/Simple-Calculator-Using-Assembly-Language. (n.d.). GitHub. https://github.com/IbrahiimKhalil/Simple-Calculator-Using-Assembly-Language/blob/master/Simple%20Calculator.asm

Simple-Calculator-Using-Assembly-Language/Simple Calculator.asm at master · IbrahiimKhalil/Simple-Calculator-Using-Assembly-Language. (n.d.). GitHub. https://github.com/IbrahiimKhalil/Simple-Calculator-Using-Assembly-Language/blob/master/Simple%20Calculator.asm

Haseeeb21/Calculator-Assembly: A Scientific Calculator coded in Assembly Language which takes numbers as input and performs the selected operations and displays the result. Operations include +, -, x, /, %, Square, Cube and some Binary Operations. (n.d.). GitHub. https://github.com/Haseeeb21/Calculator-Assembly