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 3: Parte I

Registros del CPU 8086

Para cada una de las siguientes instrucciones de ensamblador, analiza el formato y determina el modo de direccionamiento empleado.

```
Unset
1.- MOV AL, 25
2.- ADD [BX], AX
3.- SUB CX, 10
4.- MOV DL, [SI]
5.- INC DWORD PTR [ES:DI]
6.- CALL PROC_NAME
7.- CMP WORD PTR [BX + SI], 42
8.- MOV BYTE PTR [BP - 6], 128
9.- MUL WORD PTR [DI + BX + 8]
```

1. Modo de direccionamiento inmediato
2. Modo de direccionamiento Indirecto
3. Modo de direccionamiento Inmediato
4. Modo de direccionamiento Directo
5. Modo de direccionamiento Indirecto con Segmentación
6. Modo de direccionamiento Directo
7. Modo de direccionamiento Indexado con base
8. Modo de direccionamiento por Registro Base
9. Modo de direccionamiento Indexado con base

Actividad 3: Parte II

Interrupciones 10H y 21H

Investiga qué son las interrupciones 10H y 21H
Observa el siguiente código en donde se emplean diversas funciones de las interrupciones 10H y 21H, identifica la función, explica como se manda a llamar y que es lo que hace.

Código A:

```
Unset
mensaje DB 'Bienvenido;, o
MOV AH, 09H
MOV DX, OFFSET mensaje
```

```
INT 21H
```

En este código se utiliza la interrupción 21H, que lo que hace es ofrecer servicios del sistema DOS, en este caso la interrupción INT 21H, con el valor 09H en AX, entonces muestra en pantalla los carácteres que estén en la dirección DX.
Entonces la ejecución sería algo así:
Define la cadena "Bienvenido", carga el valor 09H en la dirección AH, para mostrar la cadena, carga en DX la dirección del mensaje y manda a llamar la interrupción 21H para mostrar el mensaje en pantalla.

Código B:

```
Unset
MOV AH, 02H
MOV BH, 0
INT 10H
```

En este caso la interrupción 10H con la función 02H, que básicamente es para mover el cursor en pantalla.
En BH, este registro lo usamos en general para cambiar los modos de pantalla, que en este caso en específico para poner la pantalla en modo texto.

Código C:

```
Unset
MOV AH, 06H
MOV BH, 0
MOV Al, '+'
MOV CX, 1
MOV DH, 10
MOV DL, 20
INT 10H
```

En este caso estaremos usando la interrupción 10H, y primero desplaza el índice una línea desde la fila 10 y la columna 20, después de mandar el cursor una línea arriba, llena el espacio vació con el carácter "+", que está almacenado en AL, esto solo afecta a la pantalla de texto 0, que está indicado en BH 0.

INT 10h es la forma abreviada de la interrupción 0x10. Esta interrupción controla los servicios de pantalla del PC.
Esta interrupción se utiliza básicamente para mostrar texto en la pantalla (sin llamar a la INT 21h de MS-DOS o INT 80h de linux), para cambiar a modo gráfico, para establecer la paleta de colores, etc...

Funciones soportadas:

Función	Código de función	Parámetros	Retorno
Activa Modo de video	AH=00h	AL = Modo de video	AL = Bandera del Modo de video / byte del modo del controlador de CRT
Asigna forma del cursor de modo de texto	AH=01h	CH = Fila inicial de scan, CL = Fila final de scan Normalmente una celda de carácter tiene 8 líneas de scan, 0-7. Así, CX=0607h es el cursor normal de subrayado, CX=0007h es un cursor de bloque completo. Si el bit 7 de CH es activado, esto usualmente significa "Oculta el cursor"". Así CX=2607h es un cursor invisible. Algunas tarjetas de video tienen 16 líneas de scan, 00h-0Fh. Algunas tarjetas de video no usan el bit 5 de CH. Con estas, haga Inicio > Fin (ej. CX=0706h)	
Asigna posición del cursor	AH = 02h	BH = Página, DH = Fila, DL = Columna	
Lee la posición del cursor y su tamaño	AH = 03h	BH = Página	AX = 0, CH = Inicio de la línea de scan, CL = Fin de la línea de scan, DH = Fila, DL = Columna
Lee la posición del light pen (No funciona en sistemas VGA)	AH = 04h	AH = Status (0=no disparado, 1=disparado), BX = Pixel X, CH = Pixel Y, CX = número de línea del pixel para los modos 0Fh-10h, DH = Carácter Y, DL = Carácter X	
Selecciona Página activa de la pantalla	AH = 05h	AL = Número de Página	
Scroll up window	AH = 06h	AL = Líneas de scroll (0 = Borra), BH = Atributo de las líneas en blanco	
Scroll down window	AH = 07h	AL = Líneas de scroll (0 = Borra), BH = Atributo de las líneas en blanco	
Lee carácter y atributo en la posición del cursor	AH = 08h	BH = Número de Página	AH = Color, AL = Carácter
Escribe carácter y atributo en la posición del cursor	AH = 09h	AL = Carácter, BH = Número de Página, BL = Color, CX = Número de veces para escribir el carácter	
Escribe carácter solo en la posición del cursor	AH = 0Ah	AL = Carácter, BH = Número de Página, CX = Número de veces para escribir el carácter	

Asigna color de fondo/borde	AH = 0Bh, BH = 00h	BL = Color del fondo/borde (el borde solo en modos de texto)	
Asigna paleta	AH = 0Bh, BH = 01h	BL = ID de Paleta (solo fue válido en el CGA, pero las nuevas tarjetas lo soportan en muchos o todos los modos gráficos)	
Escribe pixel gráfico	AH = 0Ch	AL = Color, BH = Página, CX = X, DX = Y	
Lee pixel gráfico	AH = 0Dh	BH = Página, CX = X, DX = Y	AL = Color
Salida de teletipo	AH = 0Eh	AL = Carácter, BL = Color (solo en modo gráfico)	
Lee modo de video actual	AH = 0Fh		AL = Modo de video
Escribe string (EGA+, lo que significa como mínimo un PC AT)	AH = 13h	AL = Modo de escritura, BH = Página, BL = Color, CX = Longitud del string, DH = Fila, DL = Columna, ES:BP = Posición del string	

Modos de Video:

Modo	Resolución	Colores	Tipo
AL = 00h	40x25	16	Texto
AL = 01h	40x25	16	Texto
AL = 02h	80x25	16	Texto
AL = 03h	80x25	16	Texto
AL = 04h	320x200	4	Gráfico
AL = 05h	320x200	4	Gráfico
AL = 06h	640x200	2	Gráfico
AL = 07h	80x25	2	Texto
AL = 0Dh	320x200	16	Gráfico
AL = 0Eh	640x200	16	Gráfico
AL = 0Fh	640x350	2	Gráfico
AL = 10h	640x350	4	Gráfico EGA 64 KB
AL = 10h	640x350	16	Gráfico EGA menor de 64 KB y VGA
AL = 11h	640x480	2	Gráfico
AL = 12h	640x480	16	Gráfico
AL = 13h	320x200	256	Gráfico
BX = 100h	640x400	256	Gráfico (SVGA)
BX = 101h	640x480	256	Gráfico (SVGA)
BX = 102h	800x600	16	Gráfico (SVGA)
BX = 103h	800x600	256	Gráfico (SVGA)
BX = 104h	1024x768	16	Gráfico (SVGA)
BX = 105h	1024x768	256	Gráfico (SVGA)
BX = 106h	1028x1024	16	Gráfico (SVGA)
BX = 107h	1028x1024	256	Gráfico (SVGA)
BX = 113h	800x600	32K	Gráfico (SVGA)
BX = 114h	800x600	64K	Gráfico (SVGA)
BX = 115h	800x600	16M	Gráfico (SVGA)

Lista de servicios de la Interrupción INT 10h:

AH = 00h	Vídeo	Establecer modo de vídeo
AH = 01h	Vídeo	Establecer el tamaño del cursor
AH = 02h	Vídeo	Posicionar el cursor
AH = 03h	Vídeo	Obtener posición y tamaño del cursor
AH = 04h	Vídeo	Obtener posición del lápiz óptico (excepto VGA)
AH = 06h	Vídeo	Subir línea
AH = 07h	Vídeo	Bajar línea
AH = 0Bh BH=00h	Vídeo	Establecer color de fondo o borde
AH = 0Bh BH=01h	Vídeo	Establecer paleta gráfica
AH = 0Ch	Vídeo	Escribir pixel gráfico
AH = 0Dh	Vídeo	Leer pixel gráfico
AH = 0Eh	Vídeo	Función TeleType (escribir caracteres en la pantalla)
AH = 0Fh	Vídeo	Obtener el modo de vídeo
AX = 1100h	Vídeo	Cambiar fuente de vídeo (Modo Texto)
AX = 4F02h	SVGA	Establecer modo de vídeo SVGA
AX = 4F03h	SVGA	Obtener modo de vídeo SVGA

Interrupción 21H:

La interrupción 21H en ensamblador (DOS interrupt int 21h) es una de las más utilizadas en el sistema operativo DOS (Disk Operating System) para realizar una amplia variedad de funciones relacionadas con la gestión de archivos, entrada/salida y operaciones del sistema.

Cuando se invoca la interrupción int 21h, el valor que se coloca en el registro AH determina qué servicio o función del sistema se va a ejecutar

Fin del Programa:

```
Unset
    INT 21H AX = 4C00H
```

Descripción: Esta rutina finalizará el programa y devolverá el control al DOS. Debe llamar a esta rutina para finalizar los programas.

Uso: Entrada: AX = 4C00H
Salida: Ninguna
Registros afectados: Ninguno

Estatus del teclado:

```
Unset
    INT 21H AH = 0BH
```

Descripción: La función de esta rutina es detectar si se ha pulsado una tecla.
Uso: Entrada: AH = 0BH
Salida: AL = FF si caracter disponible
AL = 0 si caracter no disponible
Registros afectados: AL

Entrada de un carácter desde el teclado:

```
Unset
    INT 21H AH = 8H
```

Descripción: La función de esta rutina es esperar un carácter del teclado sin escribir por pantalla y almacenarlo en el registro AL en forma de código ASCII.

Uso: Entrada: AH = 8H
Salida: AL = carácter ASCII de la tecla pulsada
Registros afectados: AL

Leer una línea del programa:

```
Unset
    INT 21H AH = 0AH
```

Descripción: La función de esta rutina es la de obtener una línea de datos del teclado (que finaliza al pulsar el retorno de carro) y almacenarlos en un área de memoria. Los caracteres son mostrados en la pantalla al ser tecleados.

Uso: Entrada: AH = 0AH contiene la dirección del segmento de memoria en el cual se almacenan los datos introducidos.
DX contiene la dirección del offset de la zona de memoria del segmento anterior en la que se almacenan los datos.
En el primer byte del área debe indicarse el máximo número de caracteres a introducir sin superar 255.
Salida: Ninguna en registro. En el segundo byte del área se almacena el número de caracteres tecleados sin contar el retorno de carro.
Registros afectados: Ninguno

Salida de un carácter por pantalla:

```
Unset
    INT 21H AH = 2H
```

Descripción: La función de esta rutina es visualizar un carácter.
Uso: Entrada: AH = 2H
DL contiene el código ASCII del carácter a visualizar.
Salida: Ninguna
Registros afectados: Ninguno

Establecer nuevo vector de interrupción:

```
Unset
    INT 21H AX = 25H
```

Descripción: Esta rutina establece un nuevo vector de interrupción.
Uso: Entrada: DS:DX Dirección de la rutina de servicio
AL: Número de la interrupción
Salida: Actualización de la tabla de vectores
Registros afectados: Ninguno

Obtiene número de interrupción:

```
Unset
    INT 21H AX = 35H
```

Descripción: Esta rutina devuelve el vector de interrupción del número de interrupción que se especifique en AL.
Uso: Entrada: AL Número de la interrupción
Salida: ES:BX Vector de la interrupción
Registros afectados: Ninguno

Posicionar el cursor:

```
Unset
     INT 10H AH = 02H
```

Entrada: DH = fila (0-24)
DL = columna (0-79)
BH = número de página

Escribir un caracter en pantalla, donde está el cursor:

```
Unset
    INT 10H AH = 0AH
```

Entradas: BH = número de página
AL = carácter a escribir

Leer carácter y atributo de la posición actual del cursor:

```
Unset
    INT 10H AH = 08H
```

Entradas: BH = número de página
Salidas: AL = carácter leído
AH = atributo del carácter leído

Escribir carácter y atributo en la posición actual del cursor:

```
Unset
    INT 10H AH = 09H
```

Entradas: BH = número de página
BL = atributo del carácter
CX = número de caracteres a escribir
AL = carácter a escribir

Servicios de la interrupción 21H:

Servicio	Descripción
00	Terminacion de Programa
01	Entrada de caracteres con eco
02	Salida de caracteres
03	Entrada auxiliar
04	Salida auxiliar
05	Salida de impresora
06	E/S directa de consola
07	Entrada de caracteres no filtrados sin eco
08	Entrada de caracteres sin eco
09	Salida de una cadena de caracteres
0A	Entrada con buffer
0B	Obtener status de entrada
0C	Borrar buffer de entrada y aceptar entrada posterior
0D	Reset de disco
0E	Asignar unidad de disco po defecto
0F	Abrir fichero
10	Cerrar fichero
11	Buscar primera coincidencia
12	Buscar siguiente coincidencia
13	Borrar fichero
14	Lectura secuencial
15	Escritura secuencial
16	Crear o truncar
17	Cambiar nombre de fichero

18	Reservado
19	Obtener unidad de disco por defecto
1A	Asignar direccion de la zona de transferencia de disco
1B	Obtener informacion de ubicacion para unidad por defecto
1C	Obtener informacion de ubicacion para unidad especificada
1D	Reservado
1E	Reservado
1F	Reservado
20	Reservado
21	Lectura aleatoria
22	Escritura aleatoria

22	Escritura aleatoria
23	Obtener tamaño de fichero
24	Asignar numero de registro aleatorio
25	Asignar vector de interrupcion
26	Crear prefijo del segmento de programa
27	Lectura aleatoria de bloque
28	Escritura aleatoria de bloque
29	Analisis (transcripcion) de nombre de fichero
2A	Obtener fecha del sistema
2B	Asignar fecha al sistema
2C	Obtener hora del sistema
2D	Asignar hora al sistema
2E	Activar flag de verificacion
2F	Obtener direccion de la zona de transferencia de disco
30	Obtener numero de version MS-DOS
31	Terminar y permanecer residente
32	Reservado
33	Obtener o activar flag de Ctrl-Break
34	Reservado
35	Obtener vector de interrupcion
36	Obtener espacio libre en disco
37	Reservado
38	Obtener o asignar pais
39	Crear subdirectorio
3A	Borrar subdirectorio
3B	Asignar directorio actual
3C	Crear o truncar fichero
3D	Abrir fichero
3E	Cerrar fichero
3F	Leer fichero o dispositivo
40	Escribir en fichero o dispositivo
41	Borrar fichero
42	Desplazar puntero del fichero
43	Obtener o asignar atributos del fichero
44	Control de gestor de dispositivo (IOCTL)
45	Duplicar handle
46	Duplicacion forzada de handle
47	Obtener directorio actual
48	Asignar memoria
49	Liberar memoria
4A	Modificar Asignacion de memoria
4B	Ejecutar programa
4C	Terminar con codigo de retorno

4D	Obtener codigo de retorno
4E	Buscar primera coincidencia
4F	Buscar siguiente coincidencia
50	Reservado
51	Reservado
52	Reservado
53	Reservado
54	Obtener flag de verificacion
55	Reservado
56	Cambiar nombre de fichero
57	Obtener o cambiar fecha y hora de fichero

58	Obtener o asignar estrategia de ubicacion
59	Obtener informacion extendida de error
5A	Crear fichero temporal
5B	Crear nuevo fichero
5C	Bloquear registros
5D	Reservado
5E	Obtener nombre de la maquina/instalacion de impresora
5F	Asignar entrada de lista
60	Reservado
61	Reservado
62	Obtener direccion del prefijo del segmento de programa
63	Obtener tabla de encabezado de bytes

Actividad 3: Parte III

Introducción a la programación en Ensamblador

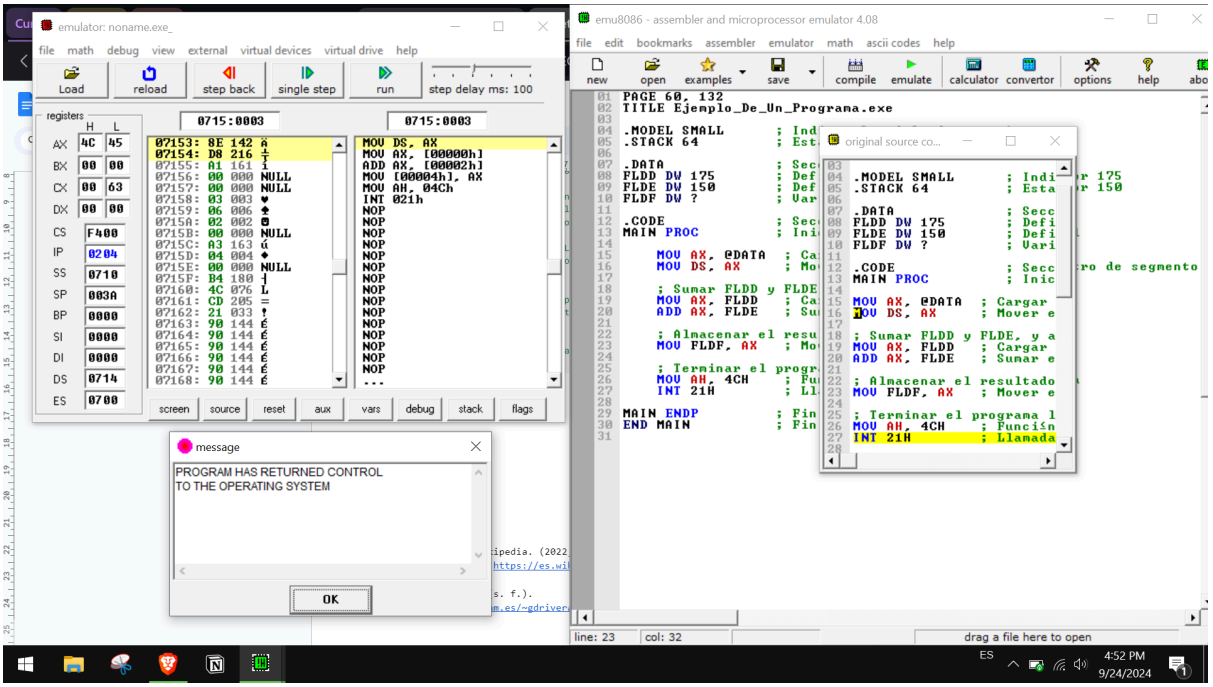
Pruebe el siguiente código en ensamblador (EMU8086) y explique lo que hace. Estructura básica de un programa en ensamblador.

```
Unset
PAGE 60, 132
TITLE Ejemplo_De_Un_Programa.exe

MODEL SMALL
STACK 64
DATA
FLDD DW 175
FLDE DW 150
FLDF DW ?
.CODE
MAIN PROC FAR
MOV AX_@DATA
MOV DS.AX
MOV AX, FLDD
ADD AX, FLDE
MOV FLDD, AX
MOV AH, HCH
INT 21H

MAIN ENDP
END MAIN
```

He probado este código en EMU8086 y me di cuenta que tiene algunos errores, pero eran errores míos, así que lo probé como debe ser y lo probé con los cambios correspondientes y a cada línea de código le puse un comentario sobre lo que hace, hay algunas que no entiendo en su totalidad, pero buscando en internet lo he corroborado, aunque siendo honesto me cuesta entender de sobremanera el Ensamblador.

```
Unset
PAGE 60, 132
```

```
TITLE Ejemplo_De_Un_Programa.exe

.MODEL SMALL       ; Indica el modelo de memoria
.STACK 64          ; Establece el tamaño de la pila

.DATA              ; Sección de datos
FLDD DW 175        ; Definición de una palabra con valor 175
FLDE DW 150        ; Definición de una palabra con valor 150
FLDF DW ?          ; Variable sin inicializar

.CODE              ; Sección de código
MAIN PROC          ; Inicio del main

    MOV AX, @DATA  ; Cargar segmento de datos en AX
    MOV DS, AX     ; Mover el valor de AX a DS

    ; Sumar FLDD y FLDE, y almacenar en AX
    MOV AX, FLDD   ; Cargar el valor de FLDD en AX
    ADD AX, FLDE   ; Sumar el valor de FLDE a AX

    ; Almacenar el resultado en FLDF
    MOV FLDF, AX   ; Mover el valor de AX a FLDF

    ; Terminar el programa
    MOV AH, 4CH    ; Terminar el programa
    INT 21H        ; Llamada a interrupción

MAIN ENDP          ; Fin del main
END MAIN           ; Fin del programa
```



Estructura básica de un programa en ensamblador.

```
Unset
.MODEL SMALL          ; Modelo de memoria (small)
.STACK 100H           ; Tamaño de la pila (256 bytes)
.DATA                 ; Sección de datos

    ; Aquí se declaran las variables y constantes
    mensaje DB 'Hola, mundo!', '$'   ; Variable de tipo cadena, termina con
$

.CODE                 ; Sección de código
START:                ; Punto de inicio del programa

    ; Inicializar el segmento de datos
    MOV AX, @DATA     ; Cargar la dirección del segmento de datos en AX
    MOV DS, AX        ; Mover la dirección del segmento a DS (registro de
segmento de datos)

    ; Aquí va el código del programa (instrucciones)

    ; Ejemplo: Mostrar mensaje en pantalla usando la función 09H de DOS
    MOV AH, 09H       ; Función 09H: Mostrar cadena de caracteres
    LEA DX, mensaje   ; Cargar la dirección del mensaje en DX
    INT 21H           ; Llamar a la interrupción DOS (21H) para mostrar la
cadena

    ; Terminar el programa y regresar al sistema operativo
    MOV AH, 4CH       ; Función de salida de DOS (terminar el programa)
    INT 21H           ; Llamar a la interrupción DOS (21H)

END START             ; Marca el final del programa y define el punto de
iniio
```

Se supone que esta es la estructura del programa en ensamblador, es lo que sale con una búsqueda, pero hay partes que no entiendo del todo así que las busqué en internet para saber qué hacía cada parte, así que lo que investigué es lo que no entiendo del todo.

Las etiquetas asignan un nombre a una instrucción. Esto permite hacer referencia a ellas en el resto del programa. Pueden tener un máximo de 31 caracteres y deben terminar en “:”.

Los comentarios permiten describir las sentencias de un programa, facilitando su comprensión. Comienzan por “;”, el ensamblador ignora el resto de la línea.

Las directivas son comandos que afectan al ensamblador, no al procesador. Se puede usar para preparar segmentos y procedimientos, definir símbolos, reservar memoria, etc.

La mayoría de las directivas no generan código objeto.
Las directivas más comunes son:
Las directivas simplificadas se utilizan para la definición de segmentos.

.MODEL para usar las directivas simplificadas es necesario incluir esta directiva que define el modelo de memoria que debe usarse. Algunos de los argumentos que puede tomar son:

- ➔ TINY: para programa con un solo segmento para datos y código (tipo .COM)
- ➔ SMALL: para programas con un solo segmento de datos (64K, incluida la pila) y otro de código (64K).
- ➔ LARGE: varios segmentos de datos y código (1Mb para cada uno).
- ➔ MEDIUM: Varios segmentos de código y 1 de datos.
- ➔ COMPACT: 1 segmento de código y varios de datos.

Con esta directiva se preparan todos los segmentos y el ensamblador reconoce, a partir de este momento, las directivas .DATA, .STACK y .CODE.

- ➔ .STACK sirve para fijar un tamaño n del segmento de pila, por defecto 1K.
- ➔ .DATA abre el segmento de datos.
- ➔ .CODE abre el segmento de código, al final código debe aparecer END.

Una vez inicializado los segmentos se permite usar los símbolos @CODE y @DATA en lugar del nombre de los segmentos de código y datos respectivamente.
Justo después de la directiva .CODE hay que inicializar el segmento de datos (ya que la directiva no genera código)

```
Unset
MOV AX, @DATA
MOV DS, AX
```

SEGMENT y ENDS definen los límites de un segmento. Las definiciones de SEGMENT deben terminar con la sentencia ENDS.

Reflexión

Creo que cada vez me va gustando más el tema de ensamblador, pero he de reconocer que me cuesta comprenderlo del todo, supongo que como no tengo buena abstracción de los registros en memoria y que no me sé de memoria las instrucciones, así como las directrices me cuesta entenderlo al 100%.
Pero me quedo con la explicación del profe.
Es verdad que programar en ensamblador es programar en el nivel más bajo de una computadora, por lo que no hay protocolos y tienes el control completo de toda la computadora, sin protocolos de protección y no estoy seguro, pero creo que así no tienes restricciones de ningún tipo.

Así que me imagino que un programador que sepa programar muy bien en ensamblador puede ser una persona peligrosa en el ámbito de la ciberseguridad. Y eso en particular me llama la atención porqué me gustaría especializarme en ciberseguridad, por lo que tengo que poner aún más atención para entender y mejorar.

Bibliografía:

colaboradores de Wikipedia. (2022, January 12). Int 10h. Wikipedia, La Enciclopedia Libre. https://es.wikipedia.org/wiki/Int_10h

Funciones del DOS. (s. f.). http://arantxa.ii.uam.es/~gdrivera/labetcii/int_dos.htm#:~:text=INT%2021H%20AX%20=%204C00H,rutina%20para%20finalizar%20los%20programas.&text=Descripci%C3%B3n:%20La%20funci%C3%B3n%20de%20esta,se%20ha%20pulsado%20una%20tecla.

Grupo de Arquitectura de Computadores y Diseño Lógico. UEX, 1997., Germán Galeano Gil & Juan A. Gómez Puildo. (1996). Tabla de interrupciones (1.a ed., Vol. 1). http://ebadillo_computacion.tripod.com/ensamblador/8086_int.pdf

Programación en ensamblador. (n.d.). https://moisesrbb.tripod.com/unidad2.htm

Universidad de Sevilla. (s. f.). Ensamblador 8086/88 (Benemérita Universidad Autónoma de Puebla, Ed.; 2.a ed., Vol. 1). https://www.cs.buap.mx/~mgonzalez/asm_mododir2.pdf