

Universidad Autónoma de Baja California
Facultad de Ciencias Químicas e Ingeniería



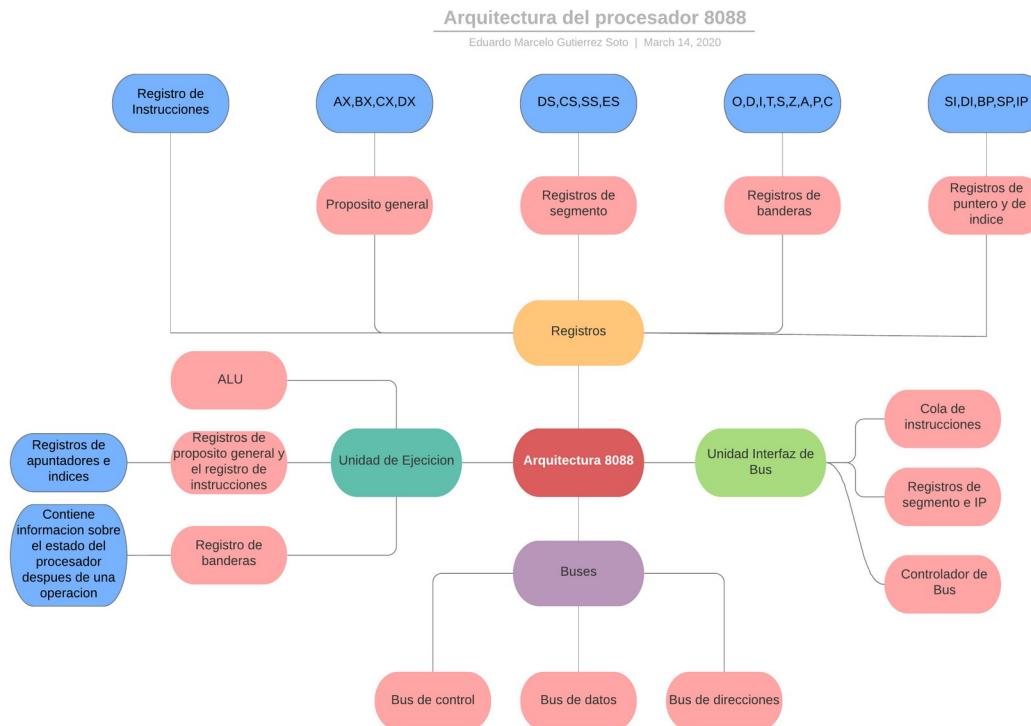
Alumno: Eduardo Marcelo Gutiérrez Soto
Profesora: Evangelina Lara Camacho
Practica 5 Modos de Direccionamiento

Objetivo

El alumno se familiarizara con los diferentes modos de direccionamiento y el conjunto de instrucciones del procesador 8086 por medio del programa debug.

Teoría

- Mapa mental sobre la arquitectura del procesador 8088.

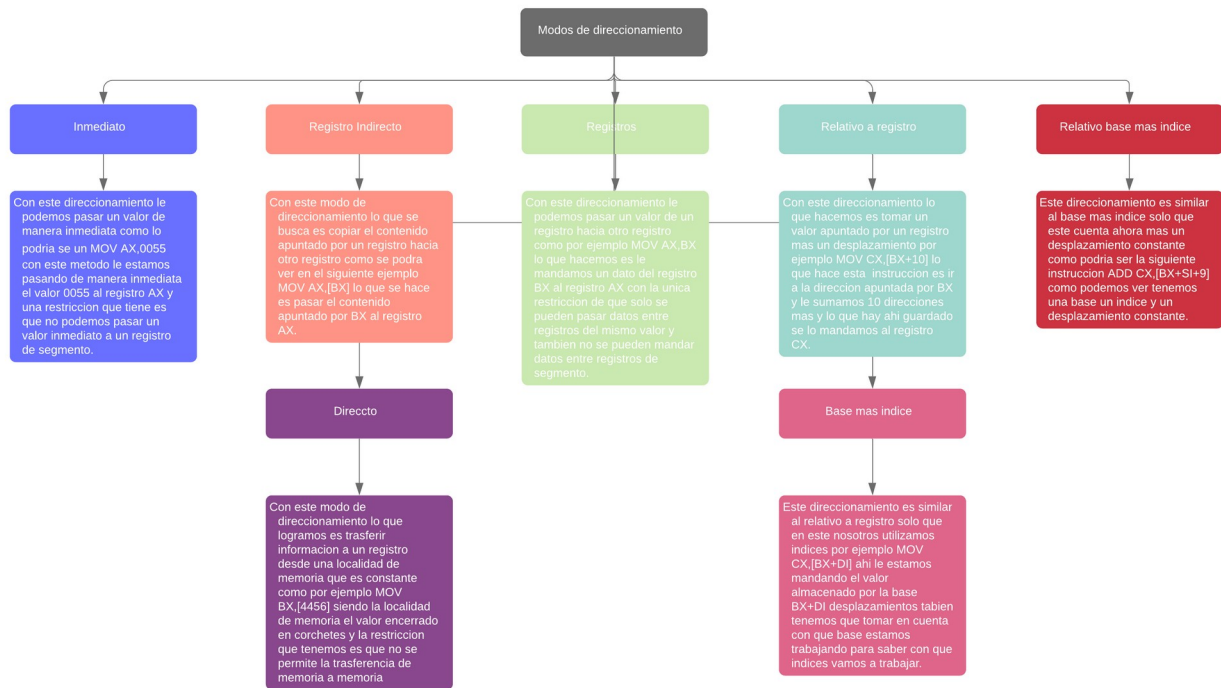


- Mapa conceptual sobre los modos de direccionamiento del 8088.

Modos de direccionamiento

Name: Eduardo marcelo gutierrez

Date: _____



- **Complete la información solicitada en la Tabla 1 sobre los comandos disponibles en el programa debug.**

Orden	Comando	Descripción	Ejemplo
Go (Ejecutar)	G <inicio><quiebre1><quiebre n>	Es un código que nos ayuda a ejecutar código de memoria si se está depurando	G<inicio> <quiebre1>
			Se inicia en una sección de memoria y en los quiebres para depurar
Hexarithmetic (Hexaritmetica)	H <valor1> <vaor2>	Este comando ejecuta sumas y restas en hexadecimal	H<FFFF><FFFD>
			Realiza una resta o suma de dos números en hexadecimal
Input (entrada)	I <puerto>	Este comando jala un byte de un puerto	I <0F>
			Saca un dato por el puerto 0F
Load (Cargar)	L <buffer><numdisco><sector ><numsector>	Este comando se utiliza para cargar un archivo o sectores de disco a memoria	
Move (Mover)	M<bloque><direccion>	Este comando mueve un bloque de memoria de una localidad a otra.	M<DS:0000><DS:00FF> <1234>
			1234 es la dirección destino
Name (Nombre)	N <NombreArchivo>	Este comando se utiliza para especificar el nombre del archivo usado por load y write	N <archivo>
			De esa forma se puede nombrar un archivo con el comando N
Output (Salida)	O <puerto><valor>	Este comando pone un byte en el puerto especificado	O <0f><1234>
			Pone en el puerto 0f un 1234
Proceed (Continuar)			
Quit (Quitar)	Q	Este comando se utiliza para salir del debug	
Trace (Trazado paso a paso)	T<inicio><cuanta>	Esta instrucción nos sirve para ejecutar paso a paso las instrucciones escritas en el debug	T<1000><5>
			Inicia en la dirección 1000h y va a ejecutar 5 instrucciones
Register (Registros)	R<registro>	Este comando despliega los registros del CPU y los valores de las banderas	
Search (Buscar)	S<bloque><valor_a_buscar>	Este comando permite buscar en un bloque de memoria una secuencia específica de valores	

Unassamble (Desensamblar)	U<alcance>	Este comando decodifica los valores de un grupo de localidades de memoria a nemonicos	U
			Y nos muestra una serie de instrucciones tecleadas.
Write (Grabar)	W<buffer><numdisk><sectorini><numsector>	Es un comando que nos sirve para escribir un archivo a sectores individuales del disc	
Assemble	-a	Se empieza a ensamblar código en una dirección de memoria en la cual IP empezara a apuntar.	Mov bx,3
			El ip estara apuntando a la instrucción almacenada en la direccion donde se empezo a ensamblar
Compare	-c	Compara un rango de memoria con otro.	
Dump	-d	Nos sirve para visualizar una seccion de memoria para los datos almacenados en dicha seccion de memoria.	-d 1234
			Va a la direccion 1234 para poder visualizar los datos que hay almacenados ahi.
Enter	-e	Comando que nos permite cambiar los contenidos de localidades de memoria en especifico	-e <4456> <hola mujer>
			Vamos a una dirección en especifico y modificamos el dato por alguno que nosotros creamos conveniente.
Fill	-f	Este comando sirve para llenar un bloque de memoria con valores esecificos	-f DS:0000 DS:00FF 0
			Llenamos desde el rengu seleccionado con el valor 0

- **Complete la información solicitada de la tabla 2 sobre el registro de banderas.**

Bandera	Descripción	Estado Activo	Estado Desactivado
Overflow(Sobre flujo)	Se activa despues de una operación de suma o resta y que a ocurrido un sobre flujo.	OV (hay overflow)	NV (no hay overflow)
Direction(Dirección)	Se selecciona el modo de auto incremento o auto decremento en operaciones con cadenas con SI y DI.	UP (hacia adelante)	DN (hacia atras)
Interrupt(Interrupción)	Habilita o deshabilita las interrupciones.	DI	EI
Sign(Signo)	Indica el signo del resultado de una operación aritmética o lógica si es negativo S=1.	PL(valor positivo)	NG (valor negativo)
Zero(Cero)	Indica el resultado de una operación aritmética y lógica es cero, en caso de que si lo sea Z=1.	ZR (es cero)	NZ (no es cero)
Auxiliary Carry(Acarreo Auxiliar)	Representa el acarreo o préstamo entre medio-bytes(nibbles) de una operación aritmética o lógica entre registros de 8 bits.	AC (hay acarreo)	NA (no hay acarreo)
Parity(Paridad)	Indica la paridad de unos en un numero resultante de una operación aritmética o lógica si es P=1 hay paridad de unos.	PE (paridad par)	PO (paridad impar)
Carry(Acarreo)	Indica un acarreo o préstamo en el bit mas significativo después de una operación aritmética esta bandera se puede modificar con algunas operaciones de corrimientos.	CY (hay acarreo)	NC (no hay acarreo)

- **Responda las siguientes preguntas.**

Que estado tiene el registro de banderas?

La bandera de overflow indica que no hay overflow.

La bandera de dirección que va hacia adelante las direcciones.

La bandera de interrupciones dice que están activadas.

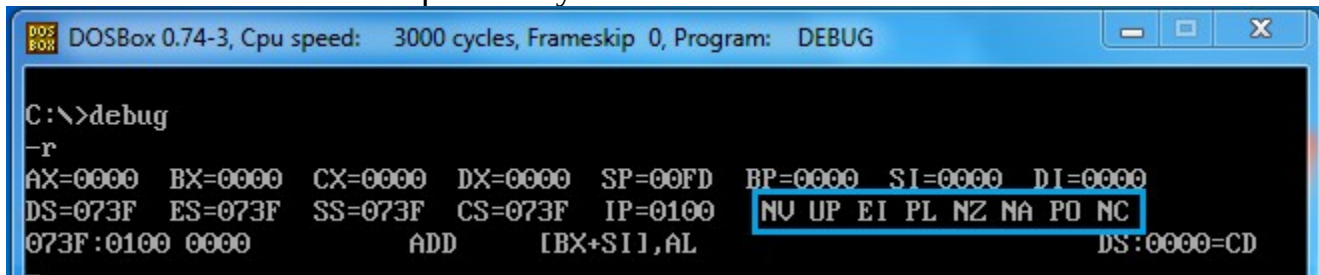
La bandera de signo dice que el numero es positivo.

La bandera de cero dice que no es cero.

La bandera de acarreo auxiliar dice que no hay acarreo auxiliar.

La bandera de paridad dice que la paridad es impar.

La bandera de acarreo dice que no hay acarreo.



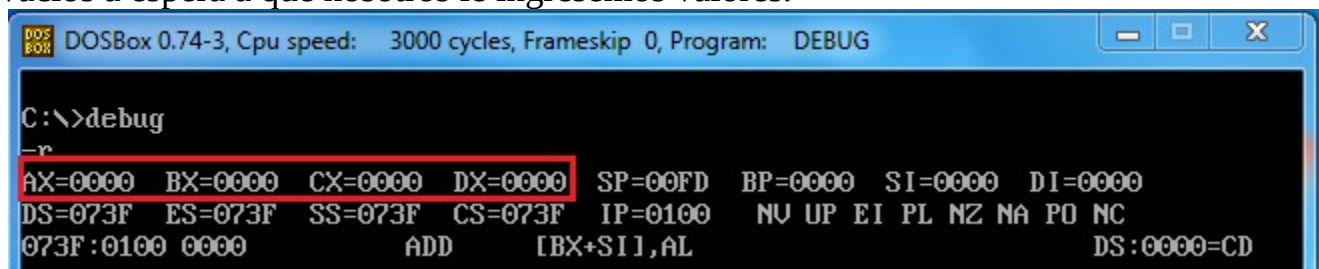
```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
C:\>debug
-r
AX=0000 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0100  NV UP EI PL NZ NA PO NC
073F:0100 0000          ADD     [BX+SI],AL          DS:0000=CD

```

Que valores toman los registros de propósito general?

El valor de los registros de propósito general se encuentran con el valor 0000 iniciando vacíos a espera a que nosotros le ingresemos valores.



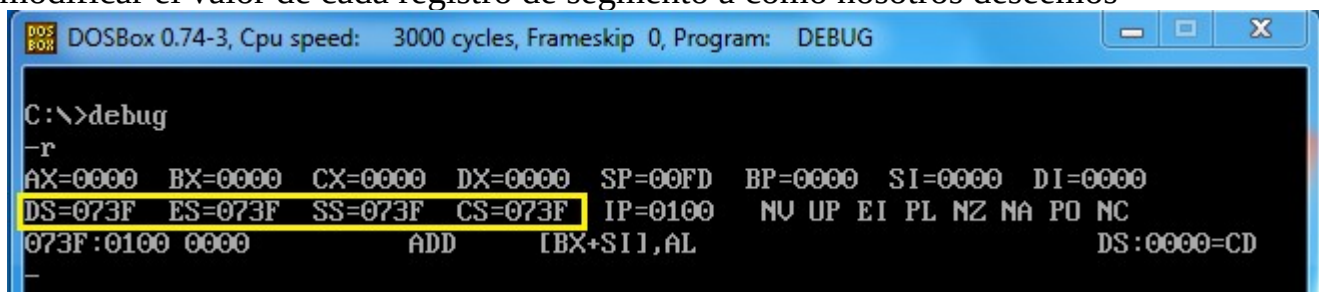
```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
C:\>debug
-r
AX=0000 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0100  NV UP EI PL NZ NA PO NC
073F:0100 0000          ADD     [BX+SI],AL          DS:0000=CD

```

El valor de los registros de segmento DS, SS, CS Y ES

los valores de los registros de segmento se encuentran compartiendo el mismo segmento de memoria el 073F pero al momento de usar el programa debug nosotros le podemos modificar el valor de cada registro de segmento a como nosotros deseemos



```

DOSBox 0.74-3, Cpu speed: 3000 cycles, Frameskip 0, Program: DEBUG
C:\>debug
-r
AX=0000 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0100  NV UP EI PL NZ NA PO NC
073F:0100 0000          ADD     [BX+SI],AL          DS:0000=CD

```

1- realice una captura de pantalla del programa debug e identifique todos los grupos de registros que cuenta el programa y así el registro de banderas.

Registros de propósito general.

Registros de puntero y de índice.

Registros de segmento

Registro de banderas

Apuntador de instrucción.

2- Utilice el programa debug para ejemplificar cada uno de los modos de direccionamiento del procesador 8088.

1. Direccionamiento a Registro.

Como podemos ver los registros ya cuentan con valores previos y la instrucción a ser ejecutada es un direccionamiento entre registros como lo que podremos ver es que le pasaremos a ah la parte de bl.

Como podemos ver hicimos una transferencia de 1 byte tenemos en la parte alta de AX el valor AC y BX queda intacto ya que no se modifica, también podemos hacer transferencias de 2 bytes como la siguiente instrucción a ejecutar MOV CX,DX

como podemos ver logramos hacer también transferencias de 2 bytes entre los registros lo que si tenemos es una limitación no podemos hacer transferencias de 1 byte a un registro completo de 2 bytes, las transferencias solo tienen que ser o de 1 byte o de 2 bytes.

2. Direcccionamiento Inmediato.

Nos permite asignarle directamente un valor a un registro como podremos ver a continuación.

```
AX=AC34 BX=FDAC CX=1235 DX=1235 SP=0013 BP=0000 SI=0009 DI=3434
DS=073F ES=073F SS=073F CS=073F IP=0119  NU UP EI PL NZ NA PO NC
073F:0119 B80000      MOV     AX,0000
```

Como podremos ver le asignaremos inmediatamente el valor 0000 al registro AX.

```
AX=0000 BX=FDAC CX=1235 DX=1235 SP=0013 BP=0000 SI=0009 DI=3434
DS=073F ES=073F SS=073F CS=073F IP=011C  NU UP EI PL NZ NA PO NC
073F:011C B738      MOV     BH,38
```

hicimos una transferencia de 2 bytes al registro AX ahora haremos una transferencia al registro BH de 1 byte.

```
AX=0000 BX=FDAC CX=1235 DX=1235 SP=0013 BP=0000 SI=0009 DI=3434
DS=073F ES=073F SS=073F CS=073F IP=011C  NU UP EI PL NZ NA PO NC
073F:011C B738      MOV     BH,38
-t
```

```
AX=0000 BX=38AC CX=1235 DX=1235 SP=0013 BP=0000 SI=0009 DI=3434
DS=073F ES=073F SS=073F CS=073F IP=011E  NU UP EI PL NZ NA PO NC
```

Como podemos observar también podemos hacer transferencias de 1 byte, ahora la única limitación que tenemos es que no podemos asignarle directamente un valor a un registro de segmento.

```
-mov ss,1234
^ Error
```

No podemos hacer transferencias inmediatas a un registro de segmento para eso necesitamos de una instrucción mas pasarle a otro registro el valor que deseamos para el segmento y de ese registro pasarlo al segmento por ejemplo.

```
073F:011E mov ss,1234
                                ^ Error
073F:011E mov ax,1234
073F:0121 mov ss,ax
073F:0123
```

De esa manera logramos modificar el valor de un registro de segmento utilizando un direccionamiento entre registros.

3. Direcccionamiento Directo.

Este direccionamiento nos sirve para acceder a una dirección de memoria y obtener el dato que se encuentra en una localidad en específico o nosotros mandar un valor a una localidad de memoria en específico con la única restricción de que no podemos hacer transferencias de memoria a memoria.

```
AX=1234 BX=38AC CX=1235 DX=1235 SP=0013 BP=0000 SI=0009 DI=3434
DS=073F ES=073F SS=1234 CS=073F IP=012C OV UP EI NG NZ NA PE NC
073F:012C 891E3412 MOV [1234],BX DS:1
```

Como podremos ver vamos a hacer una transferencia del contenido del registro BX a la dirección [1234].

```
AX=1234 BX=38AC CX=1235 DX=1235 SP=0013 BP=0000 SI=0009 DI=3434
DS=073F ES=073F SS=1234 CS=073F IP=0130 OV UP EI NG NZ NA PE NC
073F:0130 0000 ADD [BX+SI],AL DS:38B5=9C
-d 1234
073F:1230 AC 38 00 00-00 00 00 00 00 00 00 00 00 .8.....
073F:1240 00 00 00 00 00 00 00 00-00 00 00 00 00 00 .....
073F:1250 00 00 00 00 00 00 00 00-00 00 00 00 00 00 .....
073F:1260 00 00 00 00 00 00 00 00-00 00 00 00 00 00 .....
073F:1270 00 00 00 00 00 00 00 00-00 00 00 00 00 00 .....
073F:1280 00 00 00 00 00 00 00 00-00 00 00 00 00 00 .....
073F:1290 00 00 00 00 00 00 00 00-00 00 00 00 00 00 .....
073F:12A0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 .....
073F:12B0 00 00 00 00 .....
```

como podemos observar en la memoria vemos el dato de BX almacenado primero almacenamos la parte baja y después la parte alta siguiendo el orden de little endian que el dato menos significativo se guarda en la dirección menos significativa y el dato mas significativo en la dirección mas significativa y en este modo de direccionamiento la única restricción que tenemos es el direccionamiento de memoria a memoria que no es valido.

```
073F:0130 mov [1234],[5678]          ^ Error
```

4. Direccionamiento Registro indirecto.

En este direccionamiento lo que se hace es una transferencia de una localidad de memoria apuntada por un registro, accedemos a donde un registro este apuntando y lo que se encuentre en esa localidad nosotros mandarlo a otro registro.

```
AX=1234 BX=38AC CX=1235 DX=1235 SP=0013 BP=0000 SI=0009 DI=3434
DS=073F ES=073F SS=1234 CS=073F IP=0130 OV UP EI NG NZ NA PE NC
073F:0130 8B07 MOV AX,[BX] DS:3
```

Como podremos ver la siguiente instrucción mandaremos lo que este apuntado por DS x10h + BX, a el registro AX utilizamos DS ya que BX funciona con el segmento de datos y multiplicamos por 10h para poder acceder a los 20 bits de la direccion y mas BX ya que ese registro contiene el desplazamiento necesario para acceder al dato.

```
AX=FFFF BX=38AC CX=1235 DX=1235 SP=0013 BP=0000 SI=0009 DI=3434
DS=073F ES=073F SS=1234 CS=073F IP=0142 OV UP EI NG NZ NA PE NC
073F:0142 0000 ADD [BX*10],AL DS:38B5
```

Al ejecutar la instrucción vemos que el dato almacenado en AX es un FFFF, anteriormente le ingresamos el dato FFFF a la direccion apuntada por BX y ahora veremos ese dato almacenado en la direccion de BX que es la 38AC.

```
-d 38ac
073F:38A0 FF FF 00 00 .....
073F:38B0 00 00 00 00 00 9C 00 00 00 00 00 00 00 00 .....
073F:38C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
073F:38D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
073F:38E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
073F:38F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
073F:3900 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
073F:3910 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
073F:3920 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

5. Direccionamiento Base mas Indice.

En este direccionamiento se utiliza una base como lo que podria ser BX si manejamos el segmento de datos y bp si manejamos el segmento de pila ya teniendo una base a utilizar tenemos que hacer uso de los indices que usamos lo cuales pueden ser para BX,[DI,SI] y para BP,[SI,SP,DI].

```
-r
AX=FFFF BX=38AC CX=1235 DX=1235 SP=0013 BP=0000 SI=0009 DI=3434
DS=073F ES=073F SS=1234 CS=073F IP=0148 OV UP EI NG NZ NA PE NC
073F:0148 8B00 MOV AX,[BX+SI] DS:3
```

Como podemos ver la instrucción le pasaremos a AX el contenido almacenado en la direccion generada por BX+SI y el resultado quedo de la siguiente manera.

```
AX=ABCD BX=38AC CX=1235 DX=1235 SP=0013 BP=0000 SI=0009 DI=3434
DS=073F ES=073F SS=1234 CS=073F IP=014A OV UP EI NG NZ NA PE NC
073F:014A 0000 ADD [BX+SI],AL DS:38B5
```

Y el dato visualizado en memoria es el siguiente.

```
-d 38b5
073F:38B0 CD AB 00 00 00 00 00 00 00 00 00 00 .....
073F:38C0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
073F:38D0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
073F:38E0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
073F:38F0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
073F:3900 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
073F:3910 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
073F:3920 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
073F:3930 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

6. Direccionamiento Relativo a Registro.

Es relativo a registro ya que utilizamos un registro y un desplazamiento contante como lo veremos continuación en un ejemplo.

```
-r
AX=ABCD BX=38AC CX=1235 DX=1235 SP=0013 BP=0000 SI=0009 DI=3434
DS=073F ES=073F SS=1234 CS=073F IP=0150 OV UP EI NG NZ NA PE NC
073F:0150 8B4701 MOV AX,[BX+01] DS:38
```

Lo que vamos hacer es mover 2 bytes al registro AX mediante el desplazamiento de BX+1.

```
AX=1995 BX=38AC CX=1235 DX=1235 SP=0013 BP=0000 SI=0009 DI=3434
DS=073F ES=073F SS=1234 CS=073F IP=0153 OV UP EI NG NZ NA PE NC
073F:0153 0000 ADD [BX+SI],AL DS:38
```

Como podemos ver el dato que se encontraba en esa sección de memoria era el 1995 y si hacemos un desplazamiento de la memoria deberíamos de ver ese dato en la dirección correspondiente.

```
-d 38ad
073F:38A0 95 19 00 ...
073F:38B0 00 00 00 00 00 00 CD AB 00-00 00 00 00 00 00 00 00 .....
073F:38C0 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
073F:38D0 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
073F:38E0 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
073F:38F0 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
073F:3900 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
073F:3910 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
073F:3920 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
```

Como podemos ver la parte baja de color amarillo y la parte alta de color verde.

7. Direccionamiento Relativo Base mas Indice

es el mismo concepto del direccionamiento anterior solo que este consta de una base y un indice mas el desplazamiento constante, es uno de los direccionamientos mas complejos y uno de los menos usados en programas.

```
AX=1995 BX=38AC CX=1235 DX=1235 SP=0013 BP=0000 SI=0009 DI=3434
DS=073F ES=073F SS=1234 CS=073F IP=0159 OV UP EI NG NZ NA PE NC
073F:0159 8B4025 MOV AX,[BX+SI+25] DS:38
```

Como podemos ver la instrucción nosotros vamos a ir en la dirección formada por BX+SI mas 25 espacios de memoria mas adelante en azul vemos el valor actual de AX.

```
AX=2806 BX=38AC CX=1235 DX=1235 SP=0013 BP=0000 SI=0009 DI=3434
DS=073F ES=073F SS=1234 CS=073F IP=015C OV UP EI NG NZ NA PE NC
073F:015C 0000 ADD [BX+SI],AL DS:38
```

Como podemos ver en azul es el nuevo valor del registro AX y en memoria podríamos ver el dato almacenado con tan solo calcular la dirección de memoria a la que se accede.

```
-d 38da
073F:38D0 06 28 00 00 00 00 .(....
073F:38E0 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
073F:38F0 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
073F:3900 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
073F:3910 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
073F:3920 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
073F:3930 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
073F:3940 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
073F:3950 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
```

Como podemos ver en amarillo la parte baja y en rojo la parte alta del dato que se encontro en la memoria en ese direccionamiento.

3- Escriba y ejecute las instrucciones necesarias para.

- A) Almacenar en la dirección lógica DS:13h los últimos 4 dígitos de su matricula (use el valor como si fuera hexadecimal).

```
AX=0000 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0100 NU UP EI PL NZ NA PO NC
073F:0100 C70613006575 MOV WORD PTR [0013],7565 DS:001
```

Realizamos una instrucción para almacenar en memoria en la direccion [0013] una palabra que son los 4 dígitos de mi matricula.

```
-d 0013
073F:0010 65 75 01 92 01-01 01 01 00 02 FF FF FF eu.....
073F:0020 FF FF FF FF FF FF FF-FF FF FF FF 00 00 00 00 .....
073F:0030 00 00 14 00 18 00 3F 07-FF FF FF FF 00 00 00 00 .....?.....
073F:0040 05 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
073F:0050 CD 21 CB 00 00 00 00-00 00 00 00 00 20 20 20 .!.....
073F:0060 20 20 20 20 20 20 20-00 00 00 00 00 20 20 20 .....
073F:0070 20 20 20 20 20 20 20-00 00 00 00 00 00 00 00 .....
073F:0080 00 0D 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
073F:0090 00 00 00 ...
```

- B) Colocar en el acumulador su año de ingreso a UABC(use el valor como si fuera hexadecimal).

```
AX=0000 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0106 NU UP EI PL NZ NA PO NC
073F:0106 B81420 MOV AX,2014
-t
AX=2014 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0109 NU UP EI PL NZ NA PO NC
073F:0109 BA3512 MOV DX,1235
```

- C) Almacenar en la dirección lógica SS:0751h el byte mas significativo del acumulador.

```
AX=2014 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0109 NU UP EI PL NZ NA PO NC
073F:0109 88A65107 MOV [BP+0751],AH SS:0751=00
-t
```

Al ingresar el dato con bp=0000 mas 0751 accedemos al segmento de pila e ingresamos en la direccion 0751 el valor mas significativo de AX.

```
-d 0751
073F:0750 20 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0760 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0770 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0780 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:0790 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:07A0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:07B0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:07C0 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:07D0 00 ..
```

- D) Colocar el valor decimal 65,535 en el registro SI

```
AX=2014 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=FFFF DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0110 NU UP EI PL NZ NA PO NC
073F:0110 0900 OR [BX+SI],AX DS:FFFF=CD00
```

E) Inicializar el registro de segmento de datos con la dirección 1F45h.

```

AX=2014 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=FFFF DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0110  NU UP EI PL NZ NA PO NC
073F:0110 B8451F      MOV     AX,1F45
-t

AX=1F45 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=FFFF DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0113  NU UP EI PL NZ NA PO NC
073F:0113 8ED8      MOV     DS,AX
-t

AX=1F45 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=FFFF DI=0000
DS=1F45 ES=073F SS=073F CS=073F IP=0115  NU UP EI PL NZ NA PO NC
073F:0115 F0      LOCK
073F:0116 07      POP     ES

```

F) Almacenar en la dirección efectiva 1F457H del segmento de datos el valor de SI.

```

1F45:0000      FF-FF 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
1F45:0010 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
1F45:0020 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
1F45:0030 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
1F45:0040 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
1F45:0050 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
1F45:0060 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
1F45:0070 00 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
1F45:0080 00 00 00 00 00 00 00 00

```

G) Inicializar el registro de segmento de pila con la dirección 25D3H.

```

AX=1F45 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=FFFF DI=0000
DS=1F45 ES=073F SS=073F CS=073F IP=0119  NU UP EI PL NZ NA PO NC
073F:0119 B8D325      MOV     AX,25D3
-t

AX=25D3 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=FFFF DI=0000
DS=1F45 ES=073F SS=073F CS=073F IP=011C  NU UP EI PL NZ NA PO NC
073F:011C 8ED0      MOV     SS,AX
-t

AX=25D3 BX=0000 CX=0000 DX=0000 SP=00FF BP=0000 SI=FFFF DI=0000
DS=1F45 ES=0000 SS=25D3 CS=073F IP=0120  NU UP EI PL NZ NA PO NC
073F:0120 128ED0A3  ADC     CL,[BP+A3D0]      SS:A3

```

H) Almacenar en la dirección efectiva 25D49H del segmento de pila la palabra E301h.

Almacenar todo en un archivo titulado Ej3 las instrucciones usadas en este ejercicio.

4- Utilice el programa debug para ejemplificar, ejecutar y verificar el resultado de las siguientes instrucciones del procesador 8088. para cada instrucción, describa mediante los comandos e instrucciones la forma de hacer el ejemplo, la forma de ejecutarlo y la forma de verificar el resultado final.

A) Instrucciones de movimientos de datos: XCHG, IN, OUT, PUSH, POP, POPE, LAHF, SAHF, XLAT, LEA, LDS Y LES.

XCHG:

```
AX=1234 BX=5678 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0106 NU UP EI PL NZ NA PO NC
073F:0106 87C3 XCHG AX,BX
-t
AX=5678 BX=1234 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0108 NU UP EI PL NZ NA PO NC
073F:0108 0000 ADD [BX+SI],AL DS:1
```

IN:

```
AX=5678 BX=1234 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0108 NU UP EI PL NZ NA PO NC
073F:0108 E5F3 IN AX,F3
-t
AX=56FF BX=1234 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=010A NU UP EI PL NZ NA PO NC
073F:010A 0000 ADD [BX+SI],AL DS:12
```

OUT:

```
AX=56FF BX=1234 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=010A NU UP EI PL NZ NA PO NC
073F:010A E7F3 OUT F3,AX
-t
AX=56FF BX=1234 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=010C NU UP EI PL NZ NA PO NC
073F:010C 0000 ADD [BX+SI],AL DS:12
```

no se nota algun cambio ya que se supone que F3 seria un puerto de salida.

PUSH:

```
-r
AX=56FF BX=1234 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=010C NU UP EI PL NZ NA PO NC
073F:010C 50 PUSH AX
-t
AX=56FF BX=1234 CX=0000 DX=0000 SP=00FB BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=010D NU UP EI PL NZ NA PO NC
073F:010D 0000 al ADD [BX+SI],AL DS:1234=00
-d 00FB
073F:00F0 FF 56 00 00 00 .U...
073F:0100 BB 34 12 BB 78 56 87 C3-E5 F3 E7 F3 50 00 00 00 .4..xU.....P...
073F:0110 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....4...
073F:0120 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
073F:0130 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
073F:0140 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
073F:0150 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
073F:0160 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
073F:0170 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 .....
```

POP:

```

AX=56FF BX=1234 CX=0000 DX=0000 SP=00FB BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=010D  NU UP EI PL NZ NA PO NC
073F:010D 5B          POP     BX
-t

AX=56FF BX=56FF CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=010E  NU UP EI PL NZ NA PO NC
073F:010E 0000      ADD     [BX+SI],AL      DS:56FF=00
-d 00fd
073F:00F0          00 00 00
073F:0100 B8 34 12 BB 78 56 87 C3-E5 F3 E7 F3 50 5B 00 00 .4..xV.....P[.
073F:0110 00 00 00 00 00 00 00 00-00 00 00 00 34 00 2E 07 .....4...
073F:0120 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
073F:0130 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
073F:0140 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
073F:0150 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
073F:0160 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....
073F:0170 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00 .....

```

se removieron 2 bytes de la pila y se almacenaron en el registro BX.

POPF:

```

AX=56FF BX=56FF CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=010E  NU UP EI PL NZ NA PO NC
073F:010E 9D          POPF
-t

AX=56FF BX=56FF CX=0000 DX=0000 SP=00FF BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=010F  NU UP DI PL NZ NA PO NC
073F:010F 0000      ADD     [BX+SI],AL      DS:56
-a

```

LAHF:

```

AX=56FF BX=56FF CX=0000 DX=0000 SP=0101 BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0110  OV UP DI PL NZ NA PO NC
073F:0110 9F          LAHF
-t

AX=02FF BX=56FF CX=0000 DX=0000 SP=0101 BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0111  OV UP DI PL NZ NA PO NC
073F:0111 0000      ADD     [BX+SI],AL      DS:56FF=00

```

carga el byte menos significativo de las banderas en el registro AH.

SAHF:

```

AX=44FF BX=56FF CX=0000 DX=0000 SP=0101 BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0114  OV UP DI PL NZ NA PO NC
073F:0114 9E          SAHF
-t

AX=44FF BX=56FF CX=0000 DX=0000 SP=0101 BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0115  OV UP DI PL ZR NA PE NC
073F:0115 0000      ADD     [BX+SI],AL      DS:56

```

copia el contenido de AH en el byte menos significativo del registro de banderas.

XLAT:

```
AX=44FF BX=56FF CX=0000 DX=0000 SP=0101 BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0115  OV UP DI PL ZR NA PE NC
073F:0115 D7          XLAT
-t

AX=4400 BX=56FF CX=0000 DX=0000 SP=0101 BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0116  OV UP DI PL ZR NA PE NC
073F:0116 0000      ADD     [BX+SI],AL      DS:56FF=00
-d 57FE
073F:57F0          00 00
073F:5800 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:5810 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:5820 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:5830 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:5840 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:5850 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:5860 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:5870 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
```

Realiza una suma de AL con el registro BX y con ello obtiene una dirección de segmento a la cual apunta y extrae el dato y lo almacena en AL, como podemos ver almacena en AL un 00 ya que en ese segmento de memoria el 57FE no hay nada.

LEA:

```
AX=4400 BX=56FF CX=0000 DX=0000 SP=0101 BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=011A  OV UP DI PL ZR NA PE NC
073F:011A 8D07      LEA     AX,[BX]      DS:56FF=FFFF
-t

AX=56FF BX=56FF CX=0000 DX=0000 SP=0101 BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=011C  OV UP DI PL ZR NA PE NC
073F:011C 3400      XOR     AL,00
-d 56FF
073F:56F0          FF
073F:5700 FF 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:5710 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:5720 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:5730 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:5740 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:5750 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:5760 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
073F:5770 00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 .....
```

Copia la dirección a la que está apuntando BX a un registro que nosotros especifiquemos, mas no copia el contenido como podemos observar la instrucción LEA solo copio la dirección que apunta BX mas no el contenido que es FFFF.

B) Instrucciones aritméticas: ADD, ADC, INC, SUB, SBB, DEC, NEG, MUL, IMUL, DIV, IDIV, CBW Y CWD.

ADD:

```

-r
AX=56FF BX=56FF CX=0000 DX=0000 SP=0101 BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0120  OV UP DI PL ZR NA PE NC
073F:0120 01D8          ADD    AX,BX
-t

AX=ADFE BX=56FF CX=0000 DX=0000 SP=0101 BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0122  OV UP DI NG NZ AC PO NC
073F:0122 0000          ADD    [BX+SI],AL
DS:56FF

```

Realiza una suma de dos registros y como podemos ver las banderas fueron afectadas como la de signo la de cero el acarreo auxiliar el de paridad y el carry.

ADC:

```

-r
AX=ADFE BX=56FF CX=0000 DX=0000 SP=0101 BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0122  OV UP DI NG NZ AC PO NC
073F:0122 11D8          ADC    AX,BX
-t

AX=04FD BX=56FF CX=0000 DX=0000 SP=0101 BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0124  NV UP DI PL NZ AC PO CY
073F:0124 0000          ADD    [BX+SI],AL
DS:56FF

```

realizamos una suma con acarreo como podemos ver en rojo el acarreo no estaba activado y le sumo lo que hay en el acarreo que es un cero y como el resultado genero un acarreo se activo.

INC:

```

-r
AX=04FD BX=56FF CX=0000 DX=0000 SP=0101 BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0124  NV UP DI PL NZ AC PO CY
073F:0124 40          INC    AX
-t

AX=04FE BX=56FF CX=0000 DX=0000 SP=0101 BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0125  NV UP DI PL NZ NA PO CY
073F:0125 0000          ADD    [BX+SI],AL
DS:56FF

```

se incremento en 1 el valor del registro AX.

SBB:

```

-r
AX=04FE BX=56FF CX=0000 DX=0000 SP=0101 BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0125  NV UP DI PL NZ NA PO CY
073F:0125 19D8          SBB    AX,BX
-t

AX=ADFE BX=56FF CX=0000 DX=0000 SP=0101 BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0127  NV UP DI NG NZ AC PO CY
073F:0127 0000          ADD    [BX+SI],AL
DS:56FF

```

Se realizo la resta de los registros de AX y BX mas el carry y se guardo el resultado en el registro AX.

DEC:

```
-r
AX=ADFE BX=56FF CX=0000 DX=0000 SP=0101 BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0127  NU UP DI NG NZ AC PO CY
073F:0127 48          DEC    AX
-t

AX=ADFD BX=56FF CX=0000 DX=0000 SP=0101 BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0128  NU UP DI NG NZ NA PO CY
073F:0128 0000      ADD    [BX+SI],AL      DS:5
```

decrementamos en 1 el dato de AX.

NEG:

```
-r
AX=ADFD BX=56FF CX=0000 DX=0000 SP=0101 BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0128  NU UP DI NG NZ NA PO CY
073F:0128 F7D8      NEG    AX
-t

AX=5203 BX=56FF CX=0000 DX=0000 SP=0101 BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=012A  NU UP DI PL NZ AC PE CY
073F:012A 0000      ADD    [BX+SI],AL      DS:56FF=
```

realiza un complemento A2 y le suma 1 bit.

MUL:

```
-r
AX=5203 BX=56FF CX=0000 DX=0000 SP=0101 BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=012A  NU UP DI PL NZ AC PE CY
073F:012A F7E0      MUL    AX
-t
al          ah

AX=EC09 BX=56FF CX=0000 DX=1A45 SP=0101 BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=012C  OV UP DI PL NZ AC PE CY
073F:012C 0000      ADD    [BX+SI],AL      DS:56FF=
```

Realizamos la multiplicación de 16 bits y nos da un dato de 32 bits la parte baja se guarda en AX y la parte alta en DX el multiplicando se guarda en AX.

IMUL:

Es una multiplicación considerando el signo.

```
AX=EC09 BX=56FF CX=0000 DX=1A45 SP=0101 BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=012C  OV UP DI PL NZ AC PE CY
073F:012C F7E8      IMUL   AX
-t

AX=9851 BX=56FF CX=0000 DX=018E SP=0101 BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=012E  OV UP DI PL NZ AC PE CY
073F:012E 0000      ADD    [BX+SI],AL      DS:5
```

DIV:

```
AX=0013 BX=0000 CX=0010 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0104  NU UP EI PL NZ NA PO NC
073F:0104 F6F1          DIV     CL
-t
AX=0301 BX=0000 CX=0010 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0106  NU UP EI PL NZ NA PO NC
073F:0106 F1          DB      F1
```

se realizo la division con cl y en el registro AL tenemos el resultado y en AH el residuo

IDIV:

Es como la division normal solo que se respeta el signo del resultado.

```
AX=0013 BX=0000 CX=0010 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0104  NU UP EI PL NZ NA PO NC
073F:0104 F6F1          DIV     CL
-t
AX=0301 BX=0000 CX=0010 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0106  NU UP EI PL NZ NA PO NC
073F:0106 F1          DB      F1
```

CBW:

```
AX=0049 BX=000F CX=B4FB DX=0000 SP=00FD BP=0000 SI=0000 DI=5860
DS=073F ES=073F SS=073F CS=073F IP=015B  OV UP EI NG NZ AC PE NC
073F:015B 98          CBW
-t
AX=0049 BX=000F CX=B4FB DX=0000 SP=00FD BP=0000 SI=0000 DI=5860
DS=073F ES=073F SS=073F CS=073F IP=015C  OV UP EI NG NZ AC PE NC
073F:015C 0000          ADD     [BX+SI],AL          DS:000F=
```

CWD:

```
AX=0049 BX=000F CX=B4FB DX=0000 SP=00FD BP=0000 SI=0000 DI=5860
DS=073F ES=073F SS=073F CS=073F IP=015C  OV UP EI NG NZ AC PE NC
073F:015C B81DA6          MOV     AX,A61D
-t
AX=A61D BX=000F CX=B4FB DX=0000 SP=00FD BP=0000 SI=0000 DI=5860
DS=073F ES=073F SS=073F CS=073F IP=015F  OV UP EI NG NZ AC PE NC
073F:015F 99          CWD
-t
AX=A61D BX=000F CX=B4FB DX=FFFF SP=00FD BP=0000 SI=0000 DI=5860
DS=073F ES=073F SS=073F CS=073F IP=0160  OV UP EI NG NZ AC PE NC
073F:0160 0000          ADD     [BX+SI],AL          DS:000F=B8
```

C) Instrucciones logicas y de manipulacion de bits: NOT, AND, OR, XOR, TEST, SHL, SHR, SAR, ROL, ROR, RCL, RCR.

NOT:

```
-r
AX=0301 BX=0000 CX=0010 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0106 NU UP EI PL NZ NA PO NC
073F:0106 F7D0 NOT AX
-t

AX=FCFE BX=0000 CX=0010 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0108 NU UP EI PL NZ NA PO NC
073F:0108 E5F3 IN AX,F3
```

invierte los bits del numero.

AND:

realiza una operación and con un dato

```
AX=7F00 BX=000F CX=B4FB DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0125 NU UP EI PL NZ NA PO NC
073F:0125 BF61FA MOV DI,FA61
-t

AX=7F00 BX=000F CX=B4FB DX=0000 SP=00FD BP=0000 SI=0000 DI=FA61
DS=073F ES=073F SS=073F CS=073F IP=0128 NU UP EI PL NZ NA PO NC
073F:0128 81E76058 AND DI,5860
-t

AX=7F00 BX=000F CX=B4FB DX=0000 SP=00FD BP=0000 SI=0000 DI=5860
DS=073F ES=073F SS=073F CS=073F IP=012C NU UP EI PL NZ NA PE NC
073F:012C 0000 ADD [BX+SI],AL DS:000F
```

OR:

realiza una operación or

```
AX=8D5E BX=000F CX=B4FB DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=011F NU UP EI NG NZ NA PO NC
073F:011F B80057 MOV AX,5700
-t

AX=5700 BX=000F CX=B4FB DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0122 NU UP EI NG NZ NA PO NC
073F:0122 80CC28 OR AH,Z8
-t

AX=7F00 BX=000F CX=B4FB DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0125 NU UP EI PL NZ NA PO NC
073F:0125 0000 ADD [BX+SI],AL DS:00
```

TEST:

la operación test es como una and solo que no modifica el dato

```
-r
AX=FE7F BX=0000 CX=0010 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=010A NU UP EI NG NZ AC PO NC
073F:010A 85C3 TEST AX,BX
-t

AX=FE7F BX=0000 CX=0010 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=010C NU UP EI PL ZR NA PE NC
```

SHL;

```
-r
AX=FE7F BX=0000 CX=0010 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=010C  NU UP EI PL ZR NA PE NC
073F:010C D1E0          SHL     AX,1
-t

AX=FCFE BX=0000 CX=0010 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=010E  NU UP EI NG NZ AC PO CY
073F:010E 9D          POPF
```

Realiza la inserción de un cero desde la parte menos significativa del numero haciendo que los bits se muevan hacia la izquierda.

SHR:

```
AX=8D5E BX=00F0 CX=0003 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0114  OU UP EI NG NZ AC PE CY
073F:0114 B37E          MOV     BL,7E
-t

AX=8D5E BX=007E CX=0003 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0116  OU UP EI NG NZ AC PE CY
073F:0116 D2EB          SHR     BL,CL
-t

AX=8D5E BX=000F CX=0003 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0118  NU UP EI PL NZ AC PE CY
073F:0118 B202          MOV     DL,02
```

lo que hacemos con el SHR es meter ceros desde la parte mas significativa.

SAR:

lo que hace es que mantiene el bit de signo.

ROL:

```
AX=0000 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0100  NU UP EI PL NZ NA PO NC
073F:0100 B857A3          MOV     AX,A357
-t

AX=A357 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0103  NU UP EI PL NZ NA PO NC
073F:0103 B102          MOV     CL,02
-t

AX=A357 BX=0000 CX=0002 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0105  NU UP EI PL NZ NA PO NC
073F:0105 D3C0          ROL     AX,CL
-t

AX=8D5E BX=0000 CX=0002 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0107  OU UP EI PL NZ NA PO NC
073F:0107 02D3          ADD     DL,BL
```

con CL le pongo el numero de rotaciones a la izquierda que quiero realizar.

ROR:

```
AX=FCFE BX=0000 CX=0010 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=010E  NV UP EI NG NZ AC PO CY
073F:010E D1C8      ROR     AX,1
-t
AX=7E7F BX=0000 CX=0010 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0110  OV UP EI NG NZ AC PO NC
073F:0110 9F      LAHF
```

saca un dato del bit menos significativo y lo mete por el lado mas significativo y los bits se mueven hacia la derecha.

RCL:

realiza una rotación hacia la izquierda contemplando el bit de carry

RCR:

realiza una rotación a la derecha contemplando el bit del carry.

Almacenar en un archivo titulado Ej4 las instrucciones usadas en este ejercicio.

5- Escriba y ejecute en Debug las instrucciones necesarias para:

A) Colocar en el registro AX el valor 0xA357 y por medio de rotaciones obtener 0x8D5E.

```
AX=0000 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0100  NV UP EI PL NZ NA PO NC
073F:0100 B857A3      MOV     AX,A357
-t

AX=A357 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0103  NV UP EI PL NZ NA PO NC
073F:0103 B102      MOV     CL,02
-t

AX=A357 BX=0000 CX=0002 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0105  NV UP EI PL NZ NA PO NC
073F:0105 D3C0      ROL     AX,CL
-t

AX=8D5E BX=0000 CX=0002 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0107  OV UP EI PL NZ NA PO NC
073F:0107 02D3      ADD     DL,BL
```

B) Colocar en el registro BL el valor 0x7E y por medio de corrimientos obtener 0xF.

```
AX=8D5E BX=00F0 CX=0003 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0114  OV UP EI NG NZ AC PE CY
073F:0114 B37E      MOV     BL,7E
-t

AX=8D5E BX=007E CX=0003 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0116  OV UP EI NG NZ AC PE CY
073F:0116 D2EB      SHR     BL,CL
-t

AX=8D5E BX=000F CX=0003 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0118  NV UP EI PL NZ AC PE CY
073F:0118 B202      MOV     DL,02
```


- C) Colocar en el registro CX el valor 0x94F2 y por medio de enmascaramiento invertir los bits 0,3 y 13, sin modificar los demas.

```

AX=8D5E BX=000F CX=0003 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0118  NU UP EI PL NZ AC PE CY
073F:0118 B9F294      MOV     CX,94F2
-t

AX=8D5E BX=000F CX=94F2 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=011B  NU UP EI PL NZ AC PE CY
073F:011B 81F10920    XOR     CX,2009
-t

AX=8D5E BX=000F CX=B4FB DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=011F  NU UP EI NG NZ NA PO NC
073F:011F 07        POP     ES

```

- D) Colocar en el registro AH el valor 0x57 y por medio de enmascaramiento activar los bits 3 y 5, sin modificar los demas.

```

AX=8D5E BX=000F CX=B4FB DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=011F  NU UP EI NG NZ NA PO NC
073F:011F B80057      MOV     AX,5700
-t

AX=5700 BX=000F CX=B4FB DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0122  NU UP EI NG NZ NA PO NC
073F:0122 80CC28      OR      AH,Z8
-t

AX=7F00 BX=000F CX=B4FB DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0125  NU UP EI PL NZ NA PO NC
073F:0125 0000      ADD     [BX+SI],AL      DS:00

```

- E) Colocar en el registro DI el valor 0xFA61 y por medio de enmascaramiento desactivar los bits 0,9,13y15, sin

```

AX=7F00 BX=000F CX=B4FB DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0125  NU UP EI PL NZ NA PO NC
073F:0125 BF61FA      MOV     DI,FA61
-t

AX=7F00 BX=000F CX=B4FB DX=0000 SP=00FD BP=0000 SI=0000 DI=FA61
DS=073F ES=073F SS=073F CS=073F IP=0128  NU UP EI PL NZ NA PO NC
073F:0128 81E76058    AND     DI,5860
-t

AX=7F00 BX=000F CX=B4FB DX=0000 SP=00FD BP=0000 SI=0000 DI=5860
DS=073F ES=073F SS=073F CS=073F IP=012C  NU UP EI PL NZ NA PE NC
073F:012C 0000      ADD     [BX+SI],AL      DS:000F

```

modificar los demás.

- F) Colocar en el registro AL el valor 0x8E y por medio de la instrucción CBW convertirlo a una palabra que se almacene en AX respetando el signo.

La instrucción convierte CBW extiende el bit de signo de AL al registro AH. Esto conserva el signo del numero.

```

AX=FF8E BX=000F CX=B4FB DX=0000 SP=00FD BP=0000 SI=0000 DI=5860
DS=073F ES=073F SS=073F CS=073F IP=0135  OV UP EI PL NZ NA PE NC
073F:0135 B80000          MOV     AX,0000
-t

AX=0000 BX=000F CX=B4FB DX=0000 SP=00FD BP=0000 SI=0000 DI=5860
DS=073F ES=073F SS=073F CS=073F IP=0138  OV UP EI PL NZ NA PE NC
073F:0138 B08E          MOV     AL,8E
-t

AX=008E BX=000F CX=B4FB DX=0000 SP=00FD BP=0000 SI=0000 DI=5860
DS=073F ES=073F SS=073F CS=073F IP=013A  OV UP EI PL NZ NA PE NC
073F:013A 98          CBW
-t

AX=FF8E BX=000F CX=B4FB DX=0000 SP=00FD BP=0000 SI=0000 DI=5860
DS=073F ES=073F SS=073F CS=073F IP=013B  OV UP EI PL NZ NA PE NC
073F:013B 0000        ADD     [BX+SI],AL          DS:000F=

```

- G) Colocar en el registro AL el valor 0x49 y por medio de la instrucción CBW convertirlo a una palabra que se almacene en AX respetando el signo.

```

AX=0049 BX=000F CX=B4FB DX=0000 SP=00FD BP=0000 SI=0000 DI=5860
DS=073F ES=073F SS=073F CS=073F IP=015B  OV UP EI NG NZ AC PE NC
073F:015B 98          CBW
-t

AX=0049 BX=000F CX=B4FB DX=0000 SP=00FD BP=0000 SI=0000 DI=5860
DS=073F ES=073F SS=073F CS=073F IP=015C  OV UP EI NG NZ AC PE NC
073F:015C 0000        ADD     [BX+SI],AL          DS:000F=

```

- H) Colocar en el registro AX el valor 0xA61D y por medio de la instrucción CBD convertirlo a una doble palabra que se almacene en DX-AX respetando el signo.

```

AX=0049 BX=000F CX=B4FB DX=0000 SP=00FD BP=0000 SI=0000 DI=5860
DS=073F ES=073F SS=073F CS=073F IP=015C  OV UP EI NG NZ AC PE NC
073F:015C B81DA6          MOV     AX,A61D
-t

AX=A61D BX=000F CX=B4FB DX=0000 SP=00FD BP=0000 SI=0000 DI=5860
DS=073F ES=073F SS=073F CS=073F IP=015F  OV UP EI NG NZ AC PE NC
073F:015F 99          CWD
-t

AX=A61D BX=000F CX=B4FB DX=FFFF SP=00FD BP=0000 SI=0000 DI=5860
DS=073F ES=073F SS=073F CS=073F IP=0160  OV UP EI NG NZ AC PE NC
073F:0160 0000        ADD     [BX+SI],AL          DS:000F=B8

```

Es casi lo mismo solo que el bit de signo lo extiende en el registro DX y como podemos ver en DX tenemos FFFF.

- I) Colocar en el registro AX el valor 0x7320 y por medio de la instrucción CWD convertirlo a una palabra que se almacene en DX-AX respetando el signo.

```
AX=A61D BX=000F CX=B4FB DX=FFFF SP=00FD BP=0000 SI=0000 DI=5860
DS=073F ES=073F SS=073F CS=073F IP=0160 OV UP EI NG NZ AC PE NC
073F:0160 B82073 MOV AX,7320
-t
AX=7320 BX=000F CX=B4FB DX=FFFF SP=00FD BP=0000 SI=0000 DI=5860
DS=073F ES=073F SS=073F CS=073F IP=0163 OV UP EI NG NZ AC PE NC
073F:0163 99 CWD
-t
AX=7320 BX=000F CX=B4FB DX=0000 SP=00FD BP=0000 SI=0000 DI=5860
DS=073F ES=073F SS=073F CS=073F IP=0164 OV UP EI NG NZ AC PE NC
073F:0164 0000 ADD [BX+SI],AL DS:0
```

Almacenar las instrucciones en un archivo llamado Ej5.

6- Escriba y ejecute en Debug las instrucciones necesarias para:

- A) inicializar los segmentos de memoria a los rangos: segmento de datos: 3F6A0-4F69F
segmento de pila: 2BC90-3BC8F.

```
AX=0000 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0100  NU UP EI PL NZ NA PO NC
073F:0100 B86A3F          MOV     AX,3F6A
-t

AX=3F6A BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=073F ES=073F SS=073F CS=073F IP=0103  NU UP EI PL NZ NA PO NC
073F:0103 8ED8          MOV     DS,AX
-t

AX=3F6A BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=3F6A ES=073F SS=073F CS=073F IP=0105  NU UP EI PL NZ NA PO NC
073F:0105 D3C0          ROL     AX,CL
```

```
AX=3F6A BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=3F6A ES=073F SS=073F CS=073F IP=0105  NU UP EI PL NZ NA PO NC
073F:0105 B8C92B          MOV     AX,2BC9
-t

AX=2BC9 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=3F6A ES=073F SS=073F CS=073F IP=0108  NU UP EI PL NZ NA PO NC
073F:0108 8ED0          MOV     SS,AX
-t

AX=2BC9 BX=0000 CX=0000 DX=0000 SP=00FD BP=0000 SI=0000 DI=0000
DS=3F6A ES=073F SS=2BC9 CS=073F IP=010C  NU UP EI PL ZR NA PE NC
073F:010C E3BB          JCXZ   00C9
```

- B) Colocar en BX los últimos dígitos de su matricula (Tome los valores como si fueran hexadecimal), y en bp los dígitos previos. En SI almacenar el numero de semestres que esta cursando.

Conclusión

en la realización de esta practica aprendí a como utilizar la gran mayoría de instrucciones con las que cuenta el debug y logre visualizar como funcionan los modos de direccionamiento en la memoria, como se ven los datos afectados y de como nosotros poder ver como se manejan los desplazamientos de la memoria.

Dificultades

Se me presentaron una serie de dificultades a la hora de notar las restricciones bajo las que trabaja el debug lo único fue el analizar los modos de direccionamiento ya que no se puede utilizar dos indices o dos bases en un direccionamiento y también que cada instrucción tiene sus propias limitaciones.