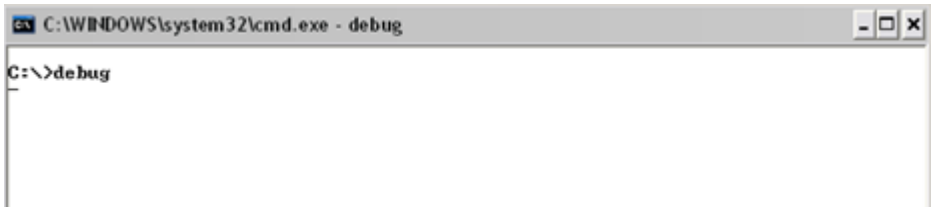
3.8. CASO DE ESTUDIO: DEBUG

Es un programa que sirve para localizar errores a través de un cierto número de técnicas interactivas, es decir, a través de la interacción con el usuario. Además de ser una valiosa herramienta de depuración, de aprendizaje utilizada para conocer paso a paso la ejecución de cada instrucción, para llevar a cabo la interacción con el usuario, el depurador cuenta con comandos que son órdenes para el programa, existe un depurador específico para cada lenguaje, para el ensamblador 8088 el programa depurador se llama ***debug***, el cual cuenta con 18 comandos que incluyen manejo de memoria de registros y de archivos, para utilizar el programa ***debug*** hay que llamarlo por su nombre y oprimir enter.

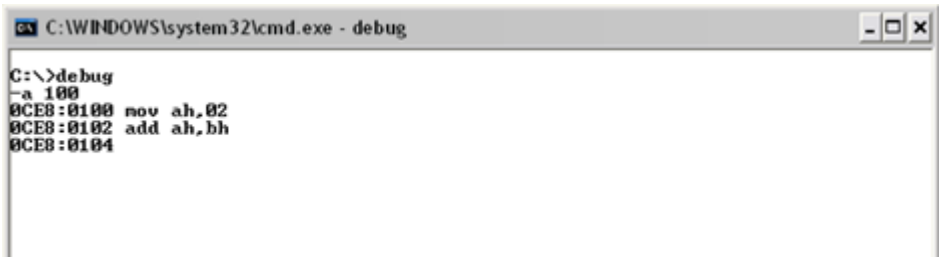


*Comandos del debug***.**

Comando A (Assemble). Se encarga de ensamblar mnemónicos 8086, 8087 y 8088, directamente en la memoria, esto es, permite meter instrucciones a partir de una dirección especificada ensamblándolas inmediatamente para ser ejecutadas. La sintaxis es:

A[dirección]

Ejemplo:



Para terminar de ensamblar solo dar enter y se retornará al prompt esperando una nueva orden.

El comando A verifica los errores de sintaxis línea a línea, si existiera un error en una línea no le permitirá al usuario escribir la siguiente línea sin haber corregido la anterior.

Ejemplo:

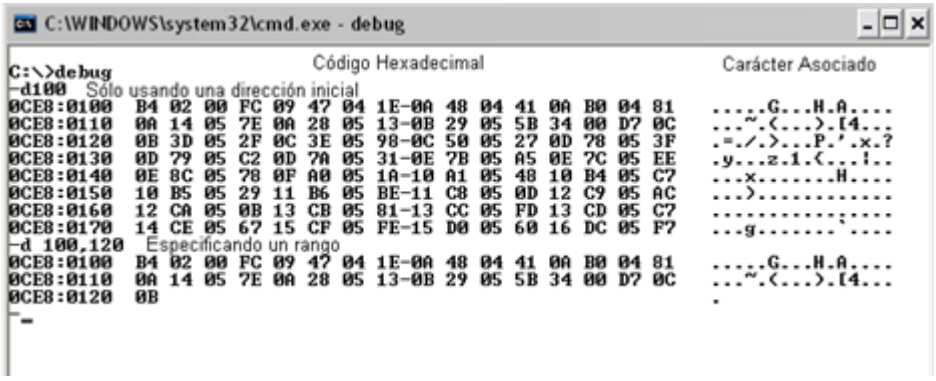
Comando D (Dump). Este comando muestra el contenido de la memoria del rango especificado en el comando, o bien muestra 128 bytes a partir de la dirección inicial especificada en el comando.

Sintaxis:

D [rango]

donde rango puede ser una sola dirección o dirección inicial, dirección final

Ejemplo:



Comando E (Enter). Muestra el contenido de la memoria permitiendo la modificación de dicho contenido. La memoria es modificada únicamente en la porción correspondiente al segmento de datos.

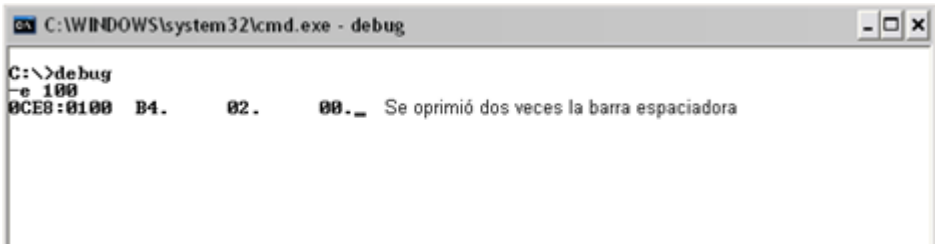
Sintaxis:

E Dirección [Lista de datos] Esta última puede omitirse

Uso: Cuando se le proporciona únicamente la dirección muestra el contenido de la memoria en esa dirección y espera cuatro posibles opciones:

1) Que el usuario oprima la barra espaciadora con lo cual mostrará el contenido de la siguiente localidad.

Ejemplo:



2) Que el usuario oprima la tecla correspondiente al guion, en cuyo caso mostrará el contenido de la localidad anterior.

Ejemplo:

3) Si el usuario oprime enter, saldrá del comando

Ejemplo:

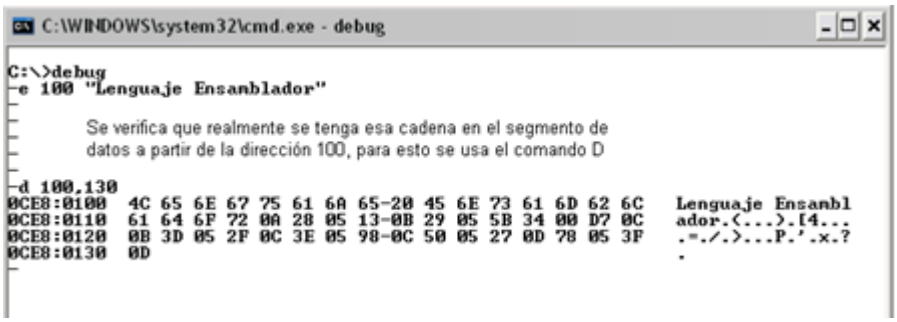
4) Modificar la información, esto es, dar un nuevo dato y oprimir enter.

Ejemplo:

En este caso en la dirección 100 en lugar de tener un 05 ahora se tendrá un 54 y en la siguiente dirección (101) en lugar de tener un 02 se tendrá ahora un 45. Para verificar esto se debe utilizar el comando D con la dirección 100.

Otra forma de usar el comando es además de dar la dirección, dar una cadena encerrada entre comillas (**“”**). Esta información será almacenada a partir de la dirección dada en el segmento de datos.

Ejemplo:

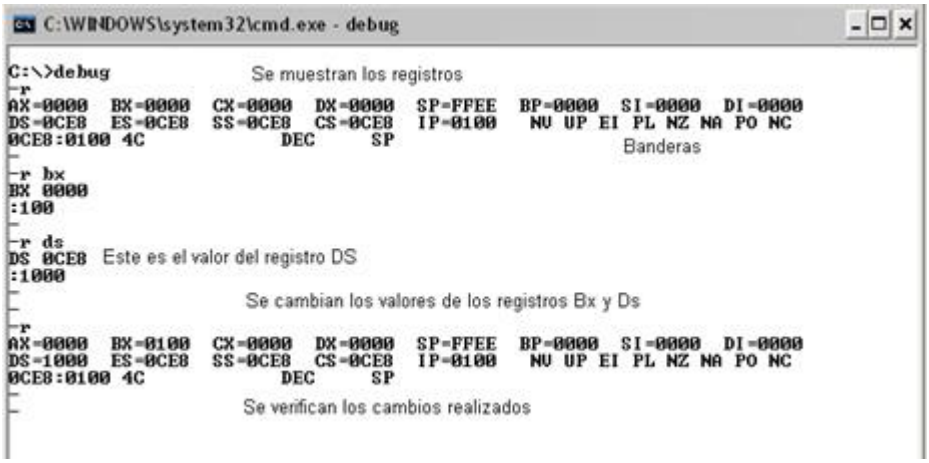


Comando R (Register). Muestra el contenido de los registros y permite modificarlos.

Sintaxis:

R [nombre del registro]

Ejemplo:



Las banderas que podemos visualizar con el comando r son 8, éstas están representadas por 2 caracteres y tienen dos estados. Aquí no aparece la bandera de Trap.

Bandera	Encendido	Apagado
OF (Overflow)	OV	NV
DF (Direction)	DN (Decremento)	UP (Incremento)
IF (Interrupt)	EI (Habilitado)	DI (Deshabilitado)
SF (Sign)	NG (Negativo)	PL (Positivo)
ZF (Zero)	ZR	NZ
AF (Auxiliary Carry)	AC	NA
PF (Parity)	PE (Par)	PO (Impar)
CF (Carry)	CY	NC

Comando T (Trace). Permite ejecutar una a una las instrucciones de un programa, a partir de la dirección especificada, o de la dirección contenida en el registro IP.

Sintaxis:

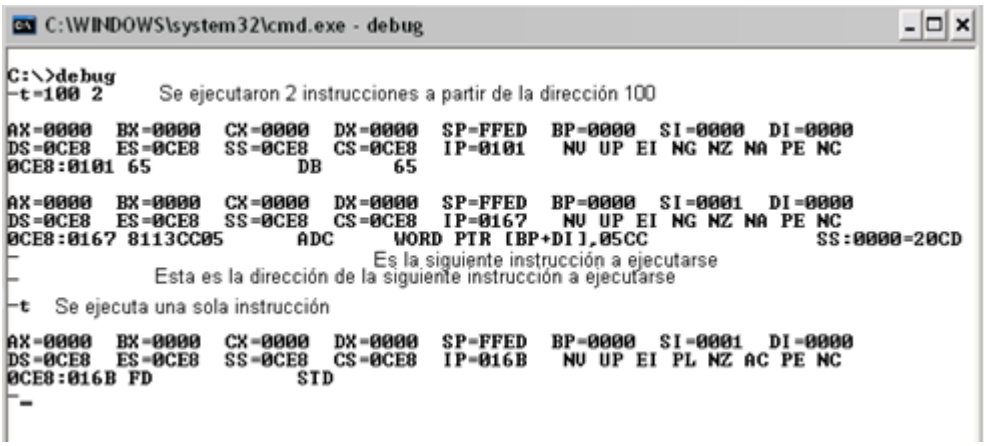
T[=dirección] [numero]

Si se omite la dirección el depurador ejecuta la instrucción cuya dirección está contenida en el registro IP.

Si no se omite el número, este comando ejecutará número de instrucciones a partir de la dirección dada o contenida en el registro IP.

Si se omite número, solo se ejecuta una sola instrucción.

Ejemplo:



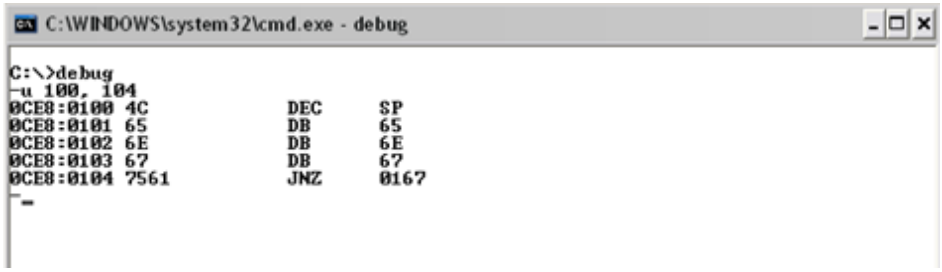
Comando U (Unassemble). Permite desensamblar o ver el contenido del segmento de código (el conjunto de instrucciones) a partir de la dirección especificada y si esta se omite, a partir de la dirección contenida en el registro IP.

Sintaxis;

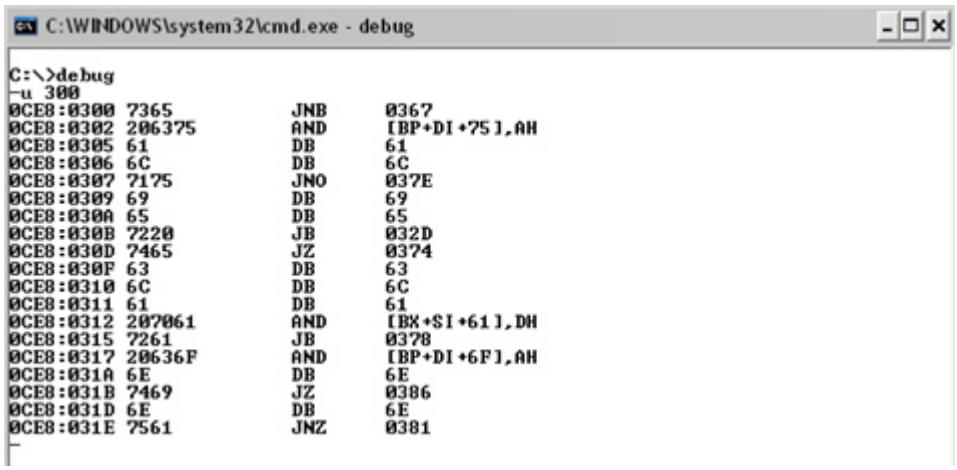
U [dirección inicial, dirección final]

Si se omite la dirección final se muestran 1F bytes del segmento de código, o menos si es que alguna instrucción sobrepasa los 1F bytes.

Ejemplo:



En el siguiente ejemplo se despliegan **1f** bytes a partir de la dirección **300**:



Comando Q (Quit). Nos permite salir del depurador.

Sintaxis:

Q

***Comando G* (Go)**. Permite ejecutar un conjunto de instrucciones.

Sintaxis:

G=dir_inicial, dir_final

Comando W (Write). Permite guardar un conjunto de instrucciones a disco.

Sintaxis:

w dir_inicial

Antes hay que colocar el número de bytes de que consta el programa en los registros BX y CX.

3.9. EJEMPLO DE ELABORACIÓN DE PROGRAMAS EN DEBUG

SUMA 2 NÚMEROS

Análisis de datos: Los datos a usar serán números enteros positivos o negativos.

Análisis de restricciones: Sin restricciones.

Algoritmo:

Suma de 2 números
Inicio

escribir("Introduzca primer número (entero): ");
leer(a);
escribir("Introduzca segundo número (entero): ");
leer(b);
suma ← a + b;
escribir(sum
a); Fin

Prueba de escritorio:

A	B	suma ← a+b	suma
2	3	2+3	5

Asociación de registros y variables:

a= ah,
b= al,
suma =ah

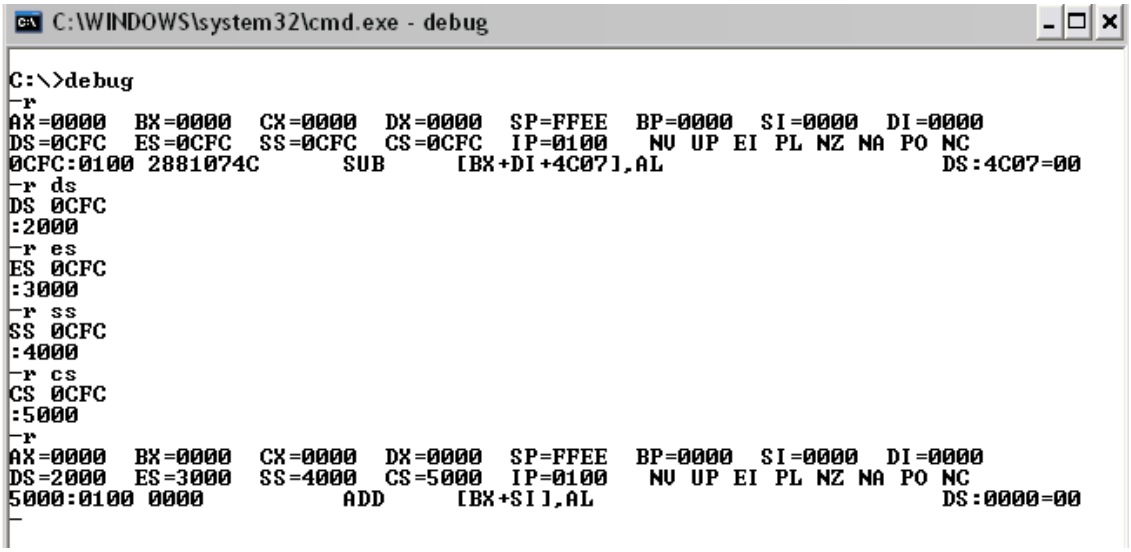
Convenciones: Suponemos que son datos válidos**.** Al ejecutar el programa se realizará la acción con los datos previamente introducidos en memoria (no sabemos leer en el lenguaje). Supongamos que hemos introducido 2 en la dirección 200 y 3 en la dirección 201. Tendremos el resultado en la dirección 202.

Código y prueba de este programa:

MOV	BX, 200	……	BX = 200 (dirección de inicio de los datos)
MOV	AH, [BX]	……	AH = 02
INC	BX	……	BX = 201
MOV	AL, [BX]	……	AL = 03
ADD	AH, AL	……	AH = 05
INC	BX	……	BX = 202
MOV	[BX], AH	……	Guarda el resultado en donde apunta BX (dirección 202).

Pasos Gráficos:

1.- Modificamos registros.



2.- Ensamblar código

```
-a 200
5000:0200 mov bx,200
5000:0203 mov ah,[bx]
5000:0205 inc bx
5000:0206 mov al,[bx]
5000:0208 add ah,al
5000:020A inc bx
5000:020B mov [bx],ah
5000:020D
```

3.-Se introducen los datos a memoria

```
-e 200
2000:0200  00.2    00.3
-
-
```

4.- Mostrar memoria

```
-
-d 200
2000:0200  02 03 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
2000:0210  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
2000:0220  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
2000:0230  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
2000:0240  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
2000:0250  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
2000:0260  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
2000:0270  00 00 00 00 00 00 00 00-00 00 00 00 00 00 00 00   ................
-
```

5.-Modificar dirección IP para la ejecución del código

```
-rip
IP 0100
:200
-
-
```

6.- Hacer ejecución con el comando t

```
-t
AX=0203  BX=0200  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=2000  ES=3000  SS=4000  CS=5000  IP=0203   NV UP EI PL NZ NA PO NC
5000:0203 8A27          MOV     AH,[BX]                         DS:0200=02
-t

AX=0203  BX=0200  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=2000  ES=3000  SS=4000  CS=5000  IP=0205   NV UP EI PL NZ NA PO NC
5000:0205 43            INC     BX
-t

AX=0203  BX=0201  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=2000  ES=3000  SS=4000  CS=5000  IP=0206   NV UP EI PL NZ NA PO NC
5000:0206 8A07          MOV     AL,[BX]                         DS:0201=03
-t

AX=0203  BX=0201  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=2000  ES=3000  SS=4000  CS=5000  IP=0208   NV UP EI PL NZ NA PO NC
5000:0208 00C4          ADD     AH,AL
-t

AX=0503  BX=0201  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=2000  ES=3000  SS=4000  CS=5000  IP=020A   NV UP EI PL NZ NA PE NC
5000:020A 43            INC     BX
-t

AX=0503  BX=0202  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=2000  ES=3000  SS=4000  CS=5000  IP=020B   NV UP EI PL NZ NA PO NC
5000:020B 8827          MOV     [BX],AH                         DS:0202=00
-t

AX=0503  BX=0202  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=2000  ES=3000  SS=4000  CS=5000  IP=020D   NV UP EI PL NZ NA PO NC
5000:020D 0000          ADD     [BX+SI],AL                      DS:0202=05
-_
```

7.- Volver a mostrar memoria

```
-d 200, 202
2000:0200  02 03 05                                          ...
-
```

3.10. EJERCICIOS

1. ¿Qué error tiene la instrucción MOV BL, CX?
2. ¿Qué error tiene la instrucción MOV DS, SS?
3. ¿Qué es una etiqueta?
4. Explique cada uno de los modos de direccionamiento
5. ¿En qué campo de una instrucción se coloca la instrucción MOV?
6. ¿Qué es un desplazamiento?, ¿Cómo determina la dirección de memoria en una instrucción MOV DS:[2000H], AL?

7. Suponga que DS = 0200H, BX=0300H y DI=400H. Determine la dirección de memoria a la que accede cada una de las siguientes instrucciones, suponiendo la operación en modo real.
    a. MOV AL, [1234h]
    b. MOV EAX, [BX]
    c. MOV [DI], AL
8. ¿Qué error tiene la siguiente instrucción MOV [BX], [DI]?
9. Explique la diferencia entre la instrucción MOV BX, DATOS y la instrucción MOV BX, OFFSET DATOS.
10. Suponga que DS = 1000H, SS=2000H, BP=1000H y DI =0100H. Determine la dirección de memoria a la que accede cada una de las siguientes instrucciones, suponiendo la operación en modo real.
    a. MOV AL, [BP+DI]
    b. MOV CX, [DI]
    c. MOV EDX, [BP]
11. ¿Qué error tiene la instrucción MOV AL, [BX][SI] si acaso hay uno?
12. Suponga que DS=1200H, BX=0100H y SI=0250H. Determine la dirección a la que accede cada una de las siguientes instrucciones, suponiendo la operación en modo real.
    a. MOV [100H], DL
    b. MOV [SI+100H], EAX
    c. MOV DL, [BX+100H
13. Suponga que DS=1100h, BX=0200H, LISTA=0250H y SI=0500H. Determine la dirección a la que accede cada una de las siguientes instrucciones, suponiendo la operación en modo real.
    a. MOV LISTA[SI], EDX
    b. MOV CL, LISTA[BX+SI]
    c. MOV CH, [BX+SI]
14. Describa la operación de cada una de las siguientes instrucciones
    a. PUSH AX
    b. POP SI
    c. PUSH [BX]
    d. POP DS
15. Compare la operación de la instrucción MOV DI, NUMERO con la instrucción LEA DI, NUMERO
16. Explique la operación de la instrucción LODSB
17. Explique la operación de la instrucción STOSW
18. ¿Qué logra el prefijo REP, y que tipo de instrucción se utiliza con él?
19. Desarrolle una secuencia de que copien 12 bytes de datos de un área de memoria direccionada por ORIGEN, hacia un área de memoria direccionada por DESTINO
20. Desarrolle un procedimiento que almacene el registro AL en cuatro posiciones de memoria consecutivas dentro del segmento de datos, cuando se direcciona mediante el registro DI.