A) GENERALIDADES E INTRODUCCIÓN AL DEBUG

1.- ¿A qué se denomina sistema operativo?

Un sistema operativo es un software encargado de brindar al usuario una forma amigable y sencilla de operar, interpretar, codificar y emitir las ordenes al procesador central para que éste realice las tareas necesarias y especificas para completar una orden. El sistema operativo es el instrumento indispensable para hacer de la computadora un objeto útil. Bajo este nombre se agrupan todos aquellos programas que permiten a los usuarios la utilización del computador, que de otra manera seria difícil de controlar. Un sistema operativo se define como un conjunto de procedimientos manuales y automáticos, que permiten a un grupo de usuarios compartir una instalación de computadora eficazmente. Es un conjunto de programas que sirven de plataforma a otros programas.

2.- Nombre por lo menos tres sistemas operativos actuales y descríbalos

Los sistemas operativos empleados actualmente son UNIX, Mac OS, Linux, MS-DOS, OS/2 y Windows: 98, 2000 (advanced server y profesional), XP, 2003.

1.- Unix

UNIX es un sistema operativo multiusuario que incorpora multitarea. Fue desarrollado originalmente por Ken Thompson y Dennis Ritchie en los laboratorios AT&T Bell en 1969 para su uso en mini computadoras. El sistema operativo UNIX tiene diversas variantes y se considera potente, más transportable e independiente de equipos concretos que otros sistemas operativos porque está escrito en lenguaje C. El UNIX está disponible en varias formas, entre las que se cuenta AIX, una versión de UNIX adaptada por IBM (para su uso en estaciones de trabajo basadas en RISC), A/UX (versión gráfica para equipos Apple Macintosh) y Mach (un sistema operativo reescrito, pero esencialmente compatible con UNIX, para las computadoras NeXT).

2.- OS/2

Es un sistema operativo multitarea para ordenadores o computadoras personales con microprocesadores de la gama x86 de Intel. El OS/2 puede ejecutar aplicaciones para Windows y MS-DOS y leer discos de MS-DOS. El OS/2 fue desarrollado originalmente como un proyecto conjunto de Microsoft e IBM.

3.- Windows

3.1 Windows NT

Es un sistema operativo diseñado para implementar principalmente redes y funcionar tanto como servidor (NT Server) como cliente (NT work station). Ofrece los mismos servicios que UNIX y puede interoperar con redes UNIX. Utiliza 32 bits y tiene compatibilidad con Windows GUI, además de soportar las aplicaciones hechas en DOS pero liberándose de las limitaciones de éste. Sus características son: extensibilidad, portabilidad, confiabilidad y robustez: compatibilidad, multiprocesamiento y escalabilidad, cómputo distribuido y desempeño, direccionamiento de 32-bits, soporte de memoria virtual, preemptive multitasking, soporte para multiprocesador y arquitectura cliente / servidor. Windows 2000 advanced server tiene básicamente las mismas características de Windows NT.

3.2 Windows XP

Windows XP Profesional integra los puntos fuertes de Windows 2000 Profesional (como la seguridad basada en estándares, la capacidad de administración y la confiabilidad) con las mejores características comerciales de Windows 98 y Windows Me (por ejemplo, Plug and Play, una interfaz de usuario más sencilla y novedosos servicios de soporte).

4. Linux

Linux es un sistema operativo muy semejante a UNIX, tanto así que casi todo el software gratuito desarrollado para UNIX se compila en Linux sin problemas. Fue diseñada por Linus Torualds en 1991 para microprocesadores 80386 y como software libre (de libre distribución), lo que le permitió un rápido avance y gran compatibilidad, aunque en la actualidad hay algunas organizaciones que la respaldan, como Red hat, Mandrake, entre otras. Linux es un sistema completo, con multitarea y multiusuario, pueden trabajar varios usuarios simultáneamente en él, y cada uno de ellos puede tener varios programas en ejecución. Soporta diversos sistemas de ficheros para guardar los datos, también soporta el ISO-9660, que es el estándar seguido en el formato de los CD-ROMs. Implementa todo lo necesario para trabajar en red con TCP/IP.

3.- ¿Qué diferencia hay entre una ROM y una RAM?

3.1 ROM (Read only memory)

Es un chip esencial de memoria de sólo lectura basada en semiconductores que contiene instrucciones o datos que "se pueden leer pero no modificar". El término ROM se suele referir a cualquier dispositivo de sólo lectura, incluyendo PROM y EPROM.

• Memoria PROM (Programmable Read-Only Memory) la memoria de sólo lectura programable puede ser escrita (programada) a través de un dispositivo especial, un programador PROM. La escritura de la memoria PROM tiene lugar fundiendo los fusibles necesarios por lo que la memoria PROM solo puede ser programada una vez.

• Memoria EPROM (Erasable Programmable Read-Only Memory) la memoria de solo lectura programable y borrable puede ser borrada mediante su exposición a la luz ultravioleta y luego se puede rescribir con un programador EPROM. Una exposición repetida a la luz ultravioleta puede destruir eventualmente la memoria EPROM pero generalmente es necesario muchas exposiciones antes de que la memoria EPROM se haga inutilizable.

• Memoria Flash o EEPROM (Electrically Erasable Read-Only Memory) la memoria de solo lectura programable y eléctricamente borrable, puede ser borrada eléctricamente y luego escrita sin sacarla del ordenador. Esta forma de escritura es más lenta que copiar en la memoria RAM o leer desde cualquier memoria ROM.

3.2 RAM (Random Access Memory)

Es una memoria de acceso aleatorio basada en semiconductores que "puede ser leída y escrita" por el microprocesador u otros dispositivos de *hardware*. Actualmente la memoria RAM para computadoras personales se suele fabricar en módulos insertables llamados DIMM, SIMM y RIMM. Tiene la capacidad de cambiar los datos de cualquier dirección en cualquier tiempo, es volátil, pues su contenido desaparece al quitarle la alimentación, almacena tanto el programa que se está ejecutando como los datos mientras están siendo procesados, almacena el sistema operativo. Se comunica con el CPU a través de los buses de dirección y el de datos. Se organiza en bites, cada uno de los cuales represent5a un valor que puede interpretarse de varias formas dependiendo de cada aplicación.

- RAM dinámica

  Almacenan la información en circuitos integrados que contienen condensadores. Como éstos pierden su carga en el transcurso del tiempo, se debe incluir los circuitos necesarios para 'refrescar' los chips de RAM. Mientras la RAM dinámica se refresca, el procesador no puede leerla. Si intenta hacerlo en ese momento, se verá forzado a esperar. Como son relativamente sencillas, las RAM dinámicas suelen utilizarse más que las RAM estáticas, a pesar de ser más lentas. Una RAM dinámica puede contener aproximadamente cuatro veces más datos que un chip de RAM estática del mismo coste.

- RAM estática

  El almacenamiento en RAM estática se basa en circuitos lógicos denominados *flip-flop*, que retienen la información almacenada en ellos mientras haya energía suficiente para hacer funcionar el dispositivo. Un chip de RAM estática puede almacenar tan sólo una cuarta parte de la información que puede almacenar un chip de RAM dinámica de la misma complejidad, pero la RAM estática no requiere ser actualizada y es normalmente mucho más rápida que la RAM dinámica. También es más cara, por lo que se reserva generalmente para su uso en la memoria de acceso aleatorio (caché).

4.- ¿Qué contiene la ROM?

Contiene el BIOS, encargado de las rutinas básicas y del control de los dispositivos de entrada y salida. El Sistema Básico de Entrada / salida (BIOS) del ROM inicia en la dirección 768k y maneja los dispositivos de entrada y salida, como un controlador de disco duro. La ROM que inicia en 960K controla las funciones básicas de la computadora, como auto prueba al encender, patrones de puntos para los gráficos y el autocargador de disco. Cuando se enciende la computadora la ROM realiza ciertas verificaciones y carga, desde el disco, los datos especiales del sistema, que envía a la RAM.

5.- ¿Qué función cumplen los registros de segmento y cuáles son?

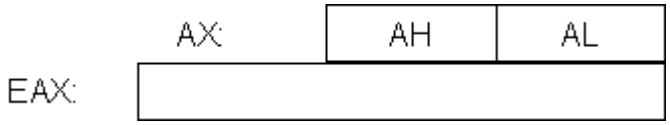
Los registros de segmento facilitan un área de memoria para direccionamiento conocida como el segmento actual y tienen 16 bits de longitud. Un segmento es un área especial en un programa que se alinea en un límite de párrafo y su dirección en un registro de segmento supone cuatro bits 0 a su derecha. Son:

a. Registro CS: Permite el direccionamiento de los segmentos de código
b. Registro DS: Genera una referencia a la localidad de un byte específico en el segmento de datos.

c. Registro SS: Permite la colocación en memoria de una pila, para almacenamiento temporal de direccionamiento y datos.
d. Registro ES: Es utilizado por algunas operaciones con cadenas de caracteres para manejar el direccionamiento de memoria.
e. Registros FS y GS: Son registros extra se segmento en los procesadores 80386 y posteriores

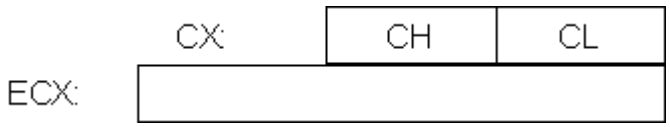
6.- ¿A qué se denomina registros de propósito general y cuál es su función?
Los registros de propósito general son "los caballos de batalla del sistema". Son únicos en el sentido de que se puede direccionarlos como una palabra o como una parte de un byte. El último byte de la izquierda es la parte "alta", y el último byte de la derecha es la parte "baja". Se puede usar los registros de propósito general para suma y resta de cifras de 8, 16 o 32 bits. Son:

- **Registro AX:** Es el acumulador principal, utilizado para operaciones que implican entrada /salida y la mayor parte de la aritmética. Por ejemplo, las instrucciones para multiplicar, dividir y traducir. También, algunas operaciones generan código mas eficiente si se refieren al AX en lugar de a los otros registros.



- **Registro BX.** El BX es conocido como el registro base ya que es el único registro de propósito general que puede ser índice para direccionamiento indexado. También es común emplear el BX para cálculos.

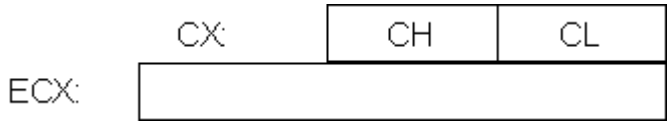
- **Registro CX**: El CX es conocido como el registro controlador. Puede contener un valor para contener el número de veces que un ciclo se repite o un valor para corrimiento de bits, hacia la derecha o hacia la izquierda. El CX también es usado para muchos cálculos.



- **Registro DX.** El DX es conocido como l registro de datos. Algunas operaciones de entrada / salida requieren uso, y las operaciones de multiplicación y división con cifras grandes suponen al DX y al AX trabajando juntos.

DX:
DL
DH

EDX:



7.- ¿De qué consta el registro de banderas y cuál es la función que cumple cada una?
El registro de banderas indica el estado actual de la computadora y los resultados de la ejecución de las instrucciones. Consta de:

a. OF (Overflow, desbordamiento) : Indica desbordamiento de un bit de orden alto (más a la izquierda) después de una operación aritmética.
b. DF (dirección) : Designa la dirección hacia la izquierda o hacia la derecha para mover o comparar cadenas de caracteres.
c. IF (interrupción) : Indica que una interrupción externa, como la entrada desde un teclado, sea procesada o ignorada.
d. TF (trampa) : Permite la operación del procesador en modo de un paso (una sola instrucción a un tiempo, para examinar el efecto de esa instrucción sobre los registros y la memoria).
e. SF (signo) : Contiene el signo resultante de una operación aritmética (0 = positivo y 1 = negativo).
f. ZF (cero) : Indica el resultado de una operación aritmética o de comparación. (0 = resultado diferente de cero y 1 = resultado igual a cero).
g. AF (acarreo auxiliar) : Contiene un acarreo externo del bit 3 en un dato de ocho bits, para aritmética especializada.
h. PF (paridad) : Indica paridad par o impar de una operación en datos de ocho bits de bajo orden (más a la derecha).
i. CF (acarreo) : Contiene el acarreo de orden más alto después de una operación aritmética, también lleva el contenido del último bit en una operación de corrimiento o de rotación.

8.- ¿Cómo es que trabaja el programa DEBUG?

El DEBUG es un programa de DOS que permite visualizar la memoria, introducir programas en ella y rastrear su ejecución. Es utilizado para probar y depurar programas ejecutables. Trabaja desplegando todo el código del programa y los datos en formato hexadecimal, y cualquier dato que se introduzca a la memoria también debe estar en formato hexadecimal. Permite también ejecutar un programa en “modo de paso sencillo”(un paso a la vez), de manera que se pueda ver el efecto de cada instrucción sobre las localidades de memoria y los registros.

9.- ¿Qué diferencia hay entre Archivos.COM y Archivos.EXE?

Algunas diferencias importantes entre un programa que es para ejecutarse como .EXE y uno que es para ejecutarse como .COM implica el tamaño del programa, la segmentación y la inicialización. Un programa .EXE puede ser de cualquier tamaño, mientras que u programa .COM está restringido a un segmento y a un máximo de 64K, incluyendo el PSP. Un programa .COM es más pequeño que su contraparte .EXE; una razón es que el bloque de encabezado de 512 bytes a un programa .EXE no precede a un programa .COM. Un programa .EXE se define con un segmento de pila y por lo común define un segmento de datos e inicializa el registro DS con la dirección de ese segmento; mientras que un programa .COM genera de manera automática una pila y sus datos están definidos dentro del segmento de código, tampoco tiene que definir el segmento de datos.

10.- ¿Cuál es la instrucción para nombrar a un programa?

Para nombrar a un programa se emplea la instrucción N seguida del nombre del archivo a nombrar, el cual tiene la extensión .com. Es decir, se escribe en el debug de la forma: N nombredearchivo.COM.

11.- ¿Cuál es la instrucción para escribir en disco un programa?

Se utiliza, desde el debug, la instrucción W (write=escribir) para escribir o grabar un programa en disco

12.- A ¿qué se denomina interrupciones?

Se denomina interrupción a una operación que suspende la ejecución de un programa de modo que el sistema pueda realizar una acción especial. La rutina de interrupción ejecuta y por lo regular regresa el control al procedimiento que fue interrumpido, el cual entonces reasume su ejecución.

13.- ¿Qué interrupciones están reservadas para el DOS?

Las interrupciones desde la 20H hasta la 3FH están reservadas para operaciones del DOS. Algunas de estas interrupciones son:

- INT 20H: Termina el programa.
- INT 21H: Petición de función al DOS.
- INT 22H: Dirección de terminación.
- INT 23H: Dirección de Cltr + Break (para transferir el control a una rutina)

- INT 24H: Manejador de error critico.
- INT 25H: Lectura absoluta de disco.
- INT 26H: Escritura absoluta de disco.
- INT 27H: Termina pero permanece residente (reside en memoria).
- INT 2FH: Interrupción de multiplexion.
- INT 33H: Manejador del ratón.

14.- ¿Cuál es la función de la Interrupción 21 y con qué registros trabaja?

Lo que realiza la interrupción 21es una petición de función al DOS. La principal operación del DOS necesita una función en el AH (trabaja con el registro de propósito general AX).

15.- ¿Cuál es la función de la interrupción 20?

Su función es terminar el programa. Finaliza la ejecución de un programa .COM, restaura las direcciones para Cltr + Break y errores críticos, limpia los bufer de registros y regresa el control al DOS. Esta función por lo regular seria colocada en el procedimiento principal y al salir de el, CS contendría la dirección del PSP. La terminación preferida es por medio de la función 4CH de la INT 21H.

16.- Indicar los comandos del Debug y la función que cumplen

Los comandos del Debug son:

- A: Ensamblar instrucciones simbólicas y pasarlas a código de máquina.
- D: Mostrar el contenido de un área de memoria.
- E: Introducir datos en memoria, iniciando en una localidad específica.
- G: Corre el programa ejecutable que se encuentra en memoria.
- N: Nombrar un programa.
- P: Proceder o ejecutar un conjunto de instrucciones relacionadas.
- Q: Salir la sesión con Debug.
- R: Mostrar el contenido de uno o más registros.
- T: Rastrear la ejecución de una instrucción.
- U: “Desensamblar” código de máquina y pasarlo a código simbólico.
- W: Escribir o grabar un programa en disco.

17.- ¿Qué significa las siglas ASCII?

Singnifica: American Standard Code for Information Interchance (Código estándar americano para el intercambio de información).

18.- ¿Cuál es la función de la instrucción CMP?

CMP = Comparar. La instrucción CMP es utilizada para comparar dos campos de datos, uno o ambos de los cuales están contenidos en un registro. El formato general para CMP es:

[Etiqueta:]	CMP	{registro/memoria},{registro/memoria/inmediato}

El resultado de una operación CMP afecta las banderas AF, CF, OF, PF, SF y ZF, aunque no es necesario probar todas estas banderas en forma individual.

CMP resta internamente el segundo operando del primero y pone en uno o en cero las banderas, pero no almacena el resultado. Ambos operandos son Byte, palabra o palabra doble (80386 y posteriores). CMP puede comparar registro, memoria o inmediato con un registro o comparar registro o inmediato con memoria.

19.- ¿A qué se denomina el bit más significativo y a qué el menos significativo?

Se denomina el bit más significativo a aquél que se encuentra más a la izquierda en un byte y es un bit menos significativo aquél que se encuentra más a la izquierda en un byte.

20.- ¿Cuál es el código ASCII?

Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec	Hex	Dec
00	000	+ 2B	043	V 56	086	ü 81	129	¼ AC	172	╫ D7	215
☺ 01	001	, 2C	044	W 57	087	é 82	130	¡ AD	173	╪ D8	216
☻ 02	002	- 2D	045	X 58	088	â 83	131	« AE	174	┘ D9	217
♥ 03	003	. 2E	046	Y 59	089	ä 84	132	» AF	175	┌ DA	218
♦ 04	004	/ 2F	047	Z 5A	090	à 85	133	░ B0	176	▉ DB	219
♣ 05	005	0 30	048	[5B	091	å 86	134	▒ B1	177	▌ DC	220
♠ 06	006	1 31	049	\ 5C	092	ç 87	135	▓ B2	178	▐ DD	221
• 07	007	2 32	050	] 5D	093	ê 88	136	│ B3	179	▀ DE	222
◘ 08	008	3 33	051	^ 5E	094	ë 89	137	┤ B4	180	DF	223
○ 09	009	4 34	052	_ 5F	095	è 8A	138	╡ B5	181	α E0	224
◙ 0A	010	5 35	053	` 60	096	ï 8B	139	╢ B6	182	ß E1	225
♂ 0B	011	6 36	054	a 61	097	î 8C	140	╖ B7	183	Γ E2	226
♀ 0C	012	7 37	055	b 62	098	ì 8D	141	╕ B8	184	π E3	227
♪ 0D	013	8 38	056	c 63	099	Ä 8E	142	╣ B9	185	Σ E4	228
♫ 0E	014	9 39	057	d 64	100	Å 8F	143	║ BA	186	σ E5	229
☼ 0F	015	: 3A	058	e 65	101	É 90	144	╗ BB	187	µ E6	230
► 10	016	; 3B	059	f 66	102	æ 91	145	╝ BC	188	τ E7	231
◄ 11	017	< 3C	060	g 67	103	Æ 92	146	╜ BD	189	Φ E8	232
↕ 12	018	= 3D	061	h 68	104	ô 93	147	╛ BE	190	Θ E9	233
‼ 13	019	> 3E	062	i 69	105	ö 94	148	┐ BF	191	Ω EA	234
¶ 14	020	? 3F	063	j 6A	106	ò 95	149	└ C0	192	δ EB	235
§ 15	021	@ 40	064	k 6B	107	û 96	150	┴ C1	193	∞ EC	236
▬ 16	022	A 41	065	l 6C	108	ù 97	151	┬ C2	194	φ ED	237
↨ 17	023	B 42	066	m 6D	109	ÿ 98	152	├ C3	195	ε EE	238
↑ 18	024	C 43	067	n 6E	110	Ö 99	153	─ C4	196	∩ EF	239
↓ 19	025	D 44	058	o 6F	111	Ü 9A	154	┼ C5	197	≡ F0	240
→ 1A	026	E 45	069	p 70	112	¢ 9B	155	╞ C6	198	± F1	241
← 1B	027	F 46	070	q 71	113	£ 9C	156	╟ C7	199	≥ F2	242
∟ 1C	028	G 47	071	r 72	114	¥ 9D	157	╚ C8	200	≤ F3	243
↔ 1D	029	H 48	072	s 73	115	₧ 9E	158	╔ C9	201	⌠ F4	244
▲ 1E	030	I 49	073	t 74	116	ƒ 9F	159	╩ CA	202	⌡ F5	245
▼ 1F	031	J 4A	074	u 75	117	á A0	160	╦ CB	203	÷ F6	246
20	032	K 4B	075	v 76	118	í A1	161	╠ CC	204	≈ F7	247
! 21	033	L 4C	076	w 77	119	ó A2	162	═ CD	205	° F8	248
" 22	034	M 4D	077	x 78	120	ú A3	163	╬ CE	206	∙ F9	249
# 23	035	N 4E	078	y 79	121	ñ A4	164	╧ CF	207	· FA	250
$ 24	036	O 4F	079	z 7A	122	Ñ A5	165	╨ D0	208	√ FB	251
% 25	037	P 50	080	{ 7B	123	ª A6	166	╤ D1	209	ⁿ FC	252
& 26	038	Q 51	081	| 7C	124	º A7	167	╥ D2	210	² FD	253
' 27	039	R 52	082	} 7D	125	¿ A8	168	╙ D3	211	■ FE	254
(28	040	S 53	083	~ 7E	126	⌐ A9	169	╘ D4	212	FF	255
) 29	041	T 54	084	⌂ 7F	127	¬ AA	170	╒ D5	213		
* 2A	042	U 55	085	Ç 80	128	½ AB	171	╓ D6	214		

21. ¿Cuál es la función de la instrucción RCL?

(RCL: Rotate left through carry = Rotar a la izquierda a través del acarreo) Su función es rotar los bits (en un byte, palabra o palabra doble) en el registro o memoria designados, hacia la izquierda a través del CF (carry flag o registro de abanderas de acarreo).

La instrucción de rotar a la izquierda se utiliza para datos lógicos y aritméticos. El operando puede ser una constante inmediata o una referencia al CL. En la 8088/86 la constante puede sólo ser 1, una rotación más grande debería estar en el CL. En procesadores posteriores, la constante puede ser hasta 31. El RCL es similar al RCR, por lo que también hace que participe el registro de banderas de acarreo en la rotación. Cada bit a la izquierda se mueve al CF y el bit del CF lo mueve a la posición vacía de la derecha. Se puede utilizar la instrucción JC para evaluar al bit que ha rotado al CF al final de una operación de rotación.

- Banderas que afecta: CF y OF
- Código fuente:

RCL {registro/memoria},{CL/constante}

C

1. **¿Cuál es la función de la instrucción LOOP?**
   Su función es controlar la ejecución de una rutina específica un determinado número de veces. El CX debe contener un dato que servirá como contador antes de comenzar las sucesivas repeticiones. El LOOP aparece al final del proceso y decrementa en uno al CX. Si el CX es diferente de cero, LOOP se transfiere a la dirección del operando, el cual apunta al comienzo del loop; caso contrario el LOOP de dirige a la instrucción siguiente. Para la 80386 y posteriores, LOOP utiliza al CX en modo de 16 bits y al ECX en modo de 32 bits.
   - Banderas que afecta: Ninguna
   - Código fuente:
     [etiqueta:] LOOP dirección

2. **¿Cuál es la función de la instrucción ADC?**
   (ADC: Add with carry = Suma con acarreo)
   Su función es ser usado en adición binaria de multipalabras para acarrear un bit 1 desbordado a la siguiente etapa de la aritmética. Suma el contenido de la bandera CF (0/1) al primer operando, y después suma el segundo operando con el primero, tal y como lo realiza ADD. Necesita 2 valores. Para sustracción de multipalabras, su equivalente es SBB.

   a. Banderas que afecta: AF, CF, OF, PF, SF y ZF
   b. Código fuente:
      ADC {registro/memoria},{registro/memoria/constante inmediata}

3. **A ¿qué se denomina acarreo y qué registro de banderas utiliza?**
   Se denomina acarreo al hecho de llevar un bit de desbordamiento de órden alto al CF de tal modo que no se pierda y se pueda trabajar de alguna u otra forma con él.
   Utiliza el registro CF (carry = acarreo), el cual contiene acarreos de alto órden (de la izquierda) siguiendo una operación aritmética y también contiene el contenido del último bit de una operación de rotación.

EJERCICIOS BÁSICOS DEBUG

Para ingresar al debug se ejecuta el DOS y se digita "debug"

1. Uso del comando "R"

Observe el contenido de los registros del microprocesador y cambie a los valores que se indican el contenido de los siguientes registros, en cada caso luego de realizar el cambio, verifíquelo en la pantalla y anote lo que va apreciando:

```
C:\>debug
- r
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
DS=1A9E  ES=1A9E  SS=1A9E  CS=1A9E  IP=0100   NV UP EI PL NZ NA PO NC
```

1.- El acumulador que se cargue con 5A7

```
-rax
:5A7
-r
AX=05A7  BX=0000  CX=0000  DX=0000
```

2.- El contador con 1DE4

```
-rcx
:1DE4
-r
AX=05A7  BX=0000  CX=1DE4  DX=0000
```

3.- El de instrucciones con 0200

```
-rip
:200
-r
(...) DS=1A9E  ES=1A9E  SS=1A9E  CS=1A9E  IP=0200
```

4.- El base con C23

```
-rbx
:C23
-r
AX=05A7  BX=0C23  CX=1DE4  DX=0000
```

5.- El DX con FFFF1

```
-rdx
:FFFF1
Error     ^
-r
AX=05A7  BX=0C23  CX=1DE4  DX=0000
```

No se puede cambiar el registro DX con FFFF1 porque tiene 2 ½ bytes (5 nibles) y el registro sólo acepta 2 bytes (4 nibles o 1 palabra).

6.- Regrese el IP a su valor original

```
-rip
:100
-r
AX=05A7  BX=0C23  CX=1DE4  DX=0000  SP=FFEE  BP=0000  SI=0000
DS=1A9E  ES=1A9E  SS=1A9E  CS=1A9E  IP=0100   NV UP EI PL NZ
```

2. Uso del comando "H"

Utilice el comando H para realizar las siguientes operaciones aritméticas con los siguientes pares de números.

a. 5D, 3D

```
-h 5D,3D
```

009A 0020

b. 3A, 11
-h3A,11
004B 0029

c. 8D, 4D
-h 8D,4D
00DA 0040

d.3AF2, B2E
-h 3AF2,B2E
4620 2FC4 7 5 2 5

1	1	1		0	1	0	1

1	0		0	1	0	1	

e. 1110101b, 100101b
-h 75,25
009A 0050 4+2+1=7 4+1= 5 2 4+1=5

f. FDE3, 234d
-h FDE3,EA 234/16: Cociente=14=E Residuo= 10=A
FECD FCF9 entonces: 234d = EA

0	1	0	1		0	1	0	1

g.0897d, 01010101b
-h 381,55 897/16: q=56 r=1
03D6 032C 4+2+1=5 4+1=5 56/16: q=3 r=8

1	0	0	0		1	0	1	0

h. 145h, 10001010b
-h 145,8A
01CF 00BB 8 8+2=10= A , entonces en hex. es 8A

i. 23456d, BCDEh 23456/16: q =1466 r = 0
-h 5BA0,BDCE 1466/16: q =91 r = 10 =A
196E 9DD2 91/16: q =5 r = 11 =B
Entonces, 23456d = 5BA0h

j. 56h, 10h
-h 56,10
0066 0046

k. A3h, 2Fh
-h A3,2F
00D2 0074

l. 4096d, 256d 16^1d = 16d = 10h
-h 1000,100 16^2d = 256d = 100h
1100 0F00 16^3d = 14096d = 1000h

3. <u>Uso del comando "E" y "T"</u>

Utilizando el lenguaje de máquina, los registros AX, BX, las instrucciones ADD (01h,d8) y SUB (29h,d8), realice las sumas y restas

3.1 <u>SUMAS</u>

Primero cargar IP con 100. Luego escribir en e100:01 y en e101:d8. En a se realizarán todos los pasos, en las siguientes se asume que previamente se han realizado los tres primeros pasos indicados anteriormente. El resultado aparece en AX..

a. 5D, 3D

```
-rip
:100
-e100
1A9E:0100  01.01
-e101
1A9E:0101  04.d8
-rax
:5D
-rbx
:3D
-t
AX=009A  BX=003D  CX=0000  DX=0000
```

b. 3A, 11

```
-rax
:3A
-rbx
:11
-t
AX=004B  BX=0011  CX=0000  DX=0000
```

3.2 RESTAS

Para las restas seguir los tres primeros pasos de b.1, es decir, cargar el IP con 100, e100 con 29 y e101 con d8.

a. 5D, 3D

```
-rip
:100
-e100
1A9E:0100  01.29
-e101
1A9E:0101  D8.d8
-rax
:5D
-rbx
:3D
-t
AX=0020  BX=003D  CX=0000  DX=0000
```

b. 3A, 11

```
-rax
:3A
-rbx
:11
-t
AX=0029  BX=0011  CX=0000  DX=0000
```

4. Uso del comando E y T : Multiplicación y división

Use las instrucciones MUL (F7h, E3h) y DIV (f7h, F3h) para realizar las siguientes operaciones.

a. 7C7C*1000 -rip
= 0707C000 :100

(DXAX)

```
-e100
1A9E:0100  29.F7
-e101
1A9E:0101  D8.E3
-rax
:7C7C
-rbx
:1000
-t
AX=C000  BX=1000  CX=0000  DX=07C7
```

b. FEAh/57h = 0048 (DHAL)

```
-rip
:100
-e100
1A9E:0100  F7.F7
-e101
1A9E:0101  F3.F3
-rax
:FEA
-rbx
:57
-t
AX=002E  BX=0057  CX=0000  DX=0048
```

5.- Muestre por la pantalla 20 caracteres ASCII (uno por uno) use el comando G y la interrupción 21 del DOS.

La interrupción 21 muestra el carácter ASCII del número hexadecimal que se pone en el registro de datos, con el puntero siempre apuntando a la dirección 100 de la memoria (pues se iniciarán las instrucciones en la dirección 10 de la memoria) y el registro acumulador cargado con 200 (en general sólo es necesario que AH esté cargado con 02 y debido a que después de mostrar un carácter el valor de AH no varía, se puede obviar). También se puede obviar el poner en memoria la interrupción 21 al mostrar varios caracteres en forma consecutiva. Para ejecutar se utiliza g102.

a. 40

```
-rip
:100
-rax
:200
-rdx
:40
-e100
85.cd   ED.21
-g102
@
```

b. 41

```
-rip
:100
-rdx
:41
-g102
A
```

6.- Repita el ejercicio anterior introduciendo ahora las instrucciones con el comando A (ensamblar), la instrucción MOV y los registros AH,DL

a. 03

```
-A
1A9E:0100 MOV AH,02
1A9E:0102 MOV DL,03
1A9E:0104 INT 21
1A9E:0106 INT 20
1A9E:0108
-G
El programa ha finalizado con normalidad
-U
```

b. C1

```
-A
1A9E:0100 MOV AH,02
1A9E:0102 MOV DL,C1
1A9E:0104 INT 21
1A9E:0106 INT 20
1A9E:0108
-G
-
El programa ha finalizado con normalidad
```

1. **Genere archivos con cada uno de los siguientes caracteres colocando como nombre EJER01.COM, EJER02.COM... y luego ejecútelos desde el DOS.**
   (En algunos se omite el BX, pues se supone que está cargado con 0000)
   -

- **01**

```
-A100
1A9E:0100 MOV AH,02
1A9E:0102 MOV DL,01
1A9E:0104 INT 21
1A9E:0106 INT 20
1A9E:0108
-N EJER01.COM
-RCX
:0008
-W
Escribiendo 00008 bytes
-Q
C:\>EJER01.COM
```

- **4E**

```
-A100
1A9E:0100 MOV AH,02
1A9E:0102 MOV DL,4E
1A9E:0104 INT 21
1A9E:0106 INT 20
1A9E:0108
-N EJER02.COM
-RCX
:0008
-W
Escribiendo 00008 bytes
-Q
C:\>EJER02.COM
N
```

-

2. **Displaye en pantalla la siguiente cadena de caracteres y guardar el programa con el nombre EJER16.COM**

```
-A100
1A9E:0100 MOV AH,09
1A9E:0102 MOV DX,200
1A9E:0105 INT 21
1A9E:0107 INT 20
1A9E:0109
-E200
1A9E:0200  4F.42   53.49   43.45   41.4E   52.56   24.45   0F.4E   BA.49
1A9E:0208  96.44   80.4F   E8.53   C3.20   1F.43   E8.41   D7.43   E2.48
1A9E:0210  BA.49   B8.4D   7E.42   E9.4F   CF.53   06.21   80.21   3E.24
-G
BIENVENIDOS CACHIMBOS!!
El programa ha finalizado con normalidad
-N EJER16.COM
-RBX
BX 0000
:0000
-RCX
CX 0000
:0124
-W
Escribiendo 00124 bytes
-Q
```

3. **Displaye en pantalla su primer nombre y guarde el programa con EJER17.COM**

```
-A100
1A9E:0100 MOV AH,09
1A9E:0102 MOV DX,200
1A9E:0105 INT 21
1A9E:0107 INT 20
1A9E:0109
-E200
1A9E:0200  80.4F   3E.53   83.43   CF.41   00.52   75.24
-G
OSCAR
El programa ha finalizado con normalidad
-N EJER17.COM
-RBX
:0000
-RCX
:0106
-W
Escribiendo 00106 bytes
-Q
```

4.- Estructure un programa con el Debug que muestre en la pantalla el siguiente logo:

```
**********************

*      CONOCIENDO      *
*    ENSAMBLADOR       *
*  ING. ELECTRONICA    *

**********************
```

10. Genere un archivo COM y ejecútelo desde el sistema operativo

```
-A100
```

```
1A9E:0100 MOV AH,09
1A9E:0102 MOV DX,200
1A9E:0105 INT 21
1A9E:0107 INT 20
1A9E:0109
-E200
1A9E:0200  80.2A   3E.2A   83.2A   CF.2A   00.2A   75.2A   0F.2A   BA.2A
1A9E:0208  96.2A   80.2A   E8.2A   C3.2A   1F.2A   E8.2A   D7.2A   E2.2A
1A9E:0210  BA.2A  B8.2A   7E.2A   E9.2A   CF.2A  06.2A   80.2A   3E.0A
1A9E:0218  8D.0D  CF.0A   00.0D   75.2A   42.20   E8.20   F1.20    03.20
1A9E:0220  72.20   10.43    80.4F   3E.4E   83.4F   CF.43   00.49    75.45
1A9E:0228  36.4E   80.44   3E.4F   C1.20   D7.20   00.20   75.20    46.20
1A9E:0230  EB.20   2D.2A  80.0A   3E.0D   9C.0A   D7.0D   00.2A   75.20
1A9E:0238  07.20   80.20   3E.20   83.20   CF.45   00.4E   74.53    08.41
1A9E:0240  C6.4D  06.42    8E.4C   CF.41   01.44    E9.4F   CA.52    FE.20
1A9E:0248  E8.20   C4.20   00.20   74.20    86.2A   80.0A   3E.0D   C2.0A
1A9E:0250  D7.0D  00.2A   74.20    03.20    E9.49    1F.4E   FF.47    C6.2E
1A9E:0258  06.20    8E.45   CF.4C   01.45   E9.43    17.54    FF.52    80.4F
1A9E:0260  3E.4E   83.49   CF.43   00.41    75.20   07.20    F6.20    06.2A
1A9E:0268  90.0A   D6.0D   02.0A   74.0D   09.2A   BA.2A   96.2A   80.2A
1A9E:0270  E8.2A   5D.2A   1F.2A   E8.2A   71.2A   E2.2A   E8.2A   B2.2A
1A9E:0278  00.2A   80.2A   3E.2A   83.2A   CF.2A   00.2A   75.2A   0A.2A
1A9E:0280  E8.2A   D2.2A   01.2A   72.24
-G
**********************

*      CONOCIENDO        *

*    ENSAMBLADOR       *

*  ING. ELECTRONICA    *

**********************
El programa ha finalizado con normalidad
-N LOGO.COM
-RBX
BX 0000
:0000
-RCX
CX 0000
:0184
-W
Escribiendo 00184 bytes
-Q
C:\>LOGO.COM
**********************
*    CONOCIENDO      *
*    ENSAMBLADOR     *
*  ING. ELECTRONICA   *

**********************
```

11. Haga un programa que imprima en pantalla el texto y realice la conversión:

NUMEROS BINARIOS
A7h = 10100111

```
-A100
1A9E:0100 MOV AH,09
```

```
1A9E:0102 MOV DX,200
1A9E:0105 INT 21
1A9E:0107 MOV AH,02
1A9E:0109 MOV BL,A7
1A9E:010B MOV CX,08
1A9E:010E RCL BL,1
1A9E:0110 MOV DL,00
1A9E:0112 ADC DL,30
1A9E:0115 INT 21
1A9E:0117 LOOP 10E
1A9E:0119 INT 20
1A9E:011B
-A200
1A9E:0200 DB "NUMEROS BINARIOS$"
1A9E:0211
-E210
1A9E:0210  26.0A   B8.0D
-A212
1A9E:0212 DB"_________________$"
1A9E:0223
-E222
1A9E:0222  26.0A   3E.0D
-A224
1A9E:0224 DB "_________________$"
1A9E:0235
-E234
1A9E:0234  24.0A   D7.0D
-E236
1A9E:0236  41.0A   37.0D
-A238
1A9E:0238 DB "A7h=$"
1A9E:023D
-G
NUMEROS BINARIOS
________________
________________

A7h=10100111
El programa ha finalizado con normalidad
```

12. <u>Utilizando la operación aritmética SUB, realice lo siguiente</u>:

a. **5 saltos de dirección con JZ**

```
-A100
1A9E:0100 MOV BL,05
1A9E:0102 SUB BL,01
1A9E:0105 JZ 109
1A9E:0107 LOOP 102
1A9E:0109 INT 20
1A9E:010B
-R
AX=0000  BX=0000  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
1A9E:0100 B305          MOV     BL,05
-T
AX=0000  BX=0005  CX=0000  DX=0000  SP=FFEE  BP=0000  SI=0000  DI=0000
1A9E:0102 80EB01        SUB     BL,01
-T
```

```
AX=0000 BX=0004 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
1A9E:0105 7402        JZ     0109
-T
AX=0000 BX=0004 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
1A9E:0107 E2F9        LOOP   0102
-T
AX=0000 BX=0004 CX=FFFF DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
1A9E:0102 80EB01      SUB    BL,01
-T
AX=0000 BX=0003 CX=FFFF DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
1A9E:0105 7402        JZ     0109
-T
AX=0000 BX=0003 CX=FFFF DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
1A9E:0107 E2F9        LOOP   0102
-T
AX=0000 BX=0003 CX=FFFE DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
1A9E:0102 80EB01      SUB    BL,01
-T
AX=0000 BX=0002 CX=FFFE DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
1A9E:0105 7402        JZ     0109
-T
AX=0000 BX=0002 CX=FFFE DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
1A9E:0107 E2F9        LOOP   0102
-T
AX=0000 BX=0002 CX=FFFD DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
1A9E:0102 80EB01      SUB    BL,01
-T
AX=0000 BX=0001 CX=FFFD DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
1A9E:0105 7402        JZ     0109
-T
AX=0000 BX=0001 CX=FFFD DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
1A9E:0107 E2F9        LOOP   0102
-T
AX=0000 BX=0001 CX=FFFC DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
1A9E:0102 80EB01      SUB    BL,01
-T
AX=0000 BX=0000 CX=FFFC DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
1A9E:0105 7402        JZ     0109
-T
AX=0000 BX=0000 CX=FFFC DX=0000 SP=FFEE BP=0000 SI=0000 DI=0000
1A9E:0109 CD20        INT    20
```

1. **Haga un programa para que imprima en pantalla todas las letras del alfabeto en minúsculas (letra a = 61h). Debe usar por lo menos un salto, también que imprima las minúsculas acentuadas (é = 130, á =160, í = 161, ó = 162, ú = 163, todos estos valores están en decimal).**

```
-A100
1A9E:0100 MOV AH,02
1A9E:0102 MOV DL,60
1A9E:0104 ADD DL,01
1A9E:0107 CMP DL,7A
1A9E:010A JA 110
1A9E:010C INT 21
1A9E:010E LOOP 104
1A9E:0110 ADD DL,07
1A9E:0113 INT 21
1A9E:0115 ADD DL,1D
```

```
1A9E:0118 MOV CX,04
1A9E:011B ADD DL,01
1A9E:011E INT 21
1A9E:0120 LOOP 11B
1A9E:0122 INT 20
1A9E:0124
-G
abcdefghijklmnopqrstuvwxyzéáíóú
El programa ha finalizado con normalidad
```

2. **Haga un programa para que imprima en pantalla todos los números hexadecimales utilizando un solo LOOP y los saltos correspondientes.**

```
-A100
1A9E:0100 MOV AH,02
1A9E:0102 MOV DL, 2F
1A9E:0104 ADD DL,01
1A9E:0107 INT 21
1A9E:0109 CMP DL,39
1A9E:010C JL 104
1A9E:010E ADD DL,07
1A9E:0111 MOV CX,06
1A9E:0114 ADD DL,01
1A9E:0117 INT 21
1A9E:0119 LOOP 114
1A9E:011B INT 20
1A9E:011D
-G
0123456789ABCDEF
El programa ha finalizado con normalidad
```

3. **Haga un programa para que imprima el dígito más significativo de los siguientes números hexadecimales:**

a.

b. **5Dh**

```
-A100
1A9E:0100 MOV DL,5D
1A9E:0102 MOV AH,02
1A9E:0104 MOV CL,4
1A9E:0106 SHR DL,CL
1A9E:0108 ADD DL,30
1A9E:010B CMP DL,39
1A9E:010E JLE 113
1A9E:0110 ADD DL,07
1A9E:0113 INT 21
1A9E:0115 INT 20
1A9E:0117
-G
5
El programa ha finalizado con normalidad
```

c. **66h**

```
-A100
1A9E:0100 MOV DL,66
1A9E:0102 MOV AH,02
1A9E:0104 MOV CL,4
1A9E:0106 SHR DL,CL
1A9E:0108 ADD DL,30
1A9E:010B CMP DL,39
1A9E:010E JLE 113
1A9E:0110 ADD DL,07
1A9E:0113 INT 21
1A9E:0115 INT 20
1A9E:0117
-G
6
El programa ha finalizado con normalidad
```

4. **Haga un programa que imprima en pantalla el dígito menos significativo de los números del ejercicio anterior.**

a.

b. **5Dh**

```
-A100
1A9E:0100 MOV DL,5D
1A9E:0102 MOV AH,02
1A9E:0104 AND DL,0F
1A9E:0107 ADD DL,30
1A9E:010A CMP DL,39
1A9E:010D JLE 112
1A9E:010F ADD DL,07
1A9E:0112 INT 21
1A9E:0114 INT 20
1A9E:0116
-G
D
El programa ha finalizado con normalidad
```

c. **66h**

```
-A100
1A9E:0100 MOV DL,66
1A9E:0102 MOV AH,02
1A9E:0104 AND DL,0F
1A9E:0107 ADD DL,30
1A9E:010A CMP DL,39
1A9E:010D JLE 112
1A9E:010F ADD DL,07
1A9E:0112 INT 21
1A9E:0114 INT 20
1A9E:0116
-G
6
El programa ha finalizado con normalidad
```

5. **Haga un programa que imprima en pantalla los dos dígitos**

a. **5Dh**

```
-A100
1A9E:0100 MOV DL,5D
1A9E:0102 MOV AH,02
1A9E:0104 MOV CL,04
1A9E:0106 SHR DL,CL
1A9E:0108 MOV CX,02
1A9E:010B ADD DL,30
1A9E:010E CMP DL,39
1A9E:0111 JLE 116
1A9E:0113 ADD DL,07
1A9E:0116 INT 21
1A9E:0118 MOV DL,5D
1A9E:011A AND DL,0F
1A9E:011D LOOP 10B
1A9E:011F INT 20
1A9E:0121
-G
5D
El programa ha finalizado con normalidad
```

b. **66h**

```
-A100
```

```
1A9E:0100 MOV DL,66
1A9E:0102 MOV AH,02
1A9E:0104 MOV CL,04
1A9E:0106 SHR DL,CL
1A9E:0108 MOV CX,02
1A9E:010B ADD DL,30
1A9E:010E CMP DL,39
1A9E:0111 JLE 116
1A9E:0113 ADD DL,07
1A9E:0116 INT 21
1A9E:0118 MOV DL,66
1A9E:011A AND DL,0F
1A9E:011D LOOP 10B
1A9E:011F INT 20
1A9E:0121
-G
66
El programa ha finalizado con normalidad
```