UNIVERSIDAD DE EL SALVADOR
FACULTAD DE INGENIERIA Y ARQUITECTURA
ESCUELA DE INGENIERIA DE SISTEMAS INFORMATICOS
ARQUITECTURA DE COMPUTADORAS.

LABORATORIO.
LA TARJETA MADRE Y SUS COMPONENTES.

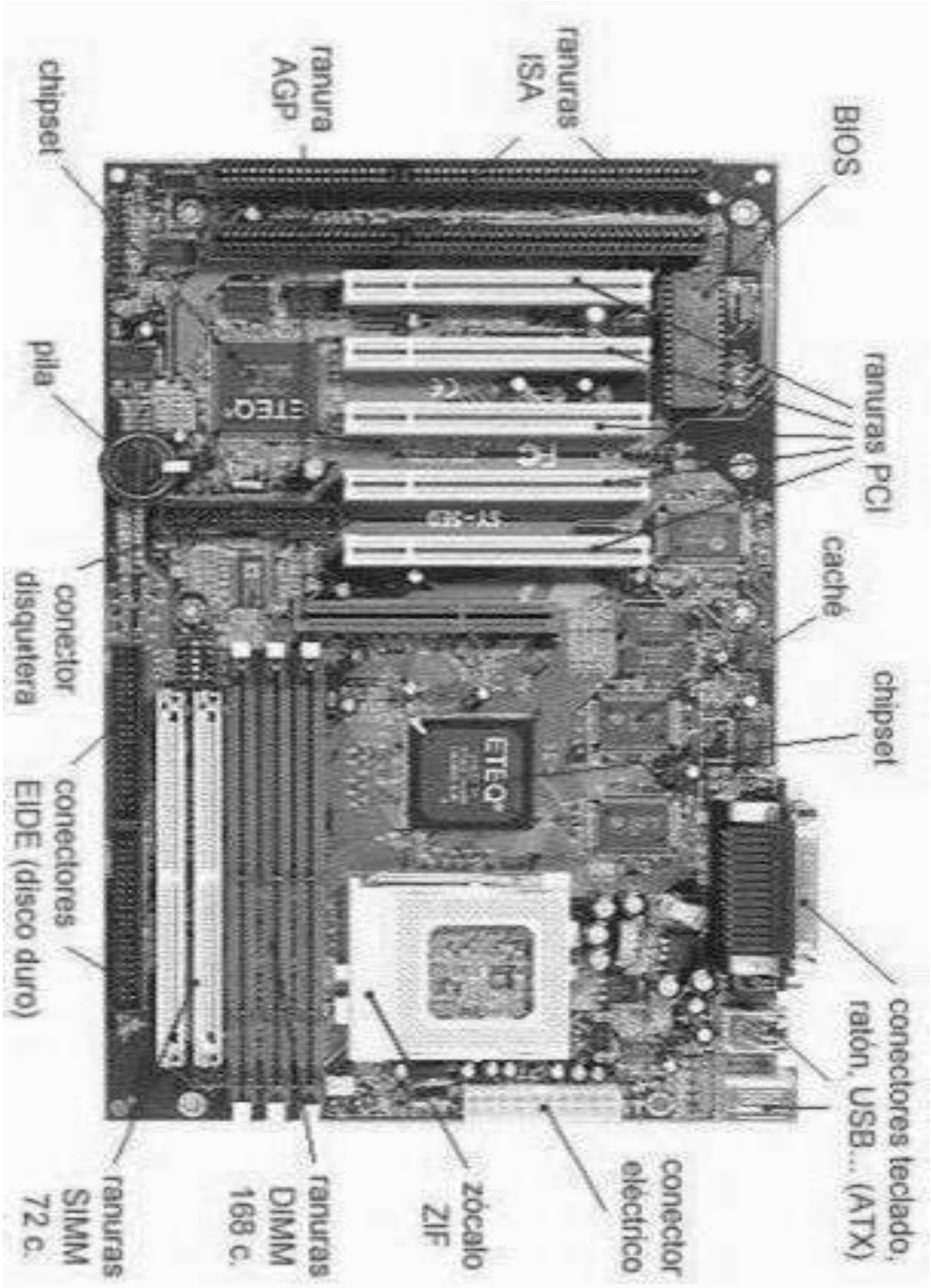
OBJETIVOS DE LA PRÁCTICA:

- Saber que es una tarjeta madre, conocerla físicamente, identificar cada una de sus partes, así como las funciones de cada una de estas.

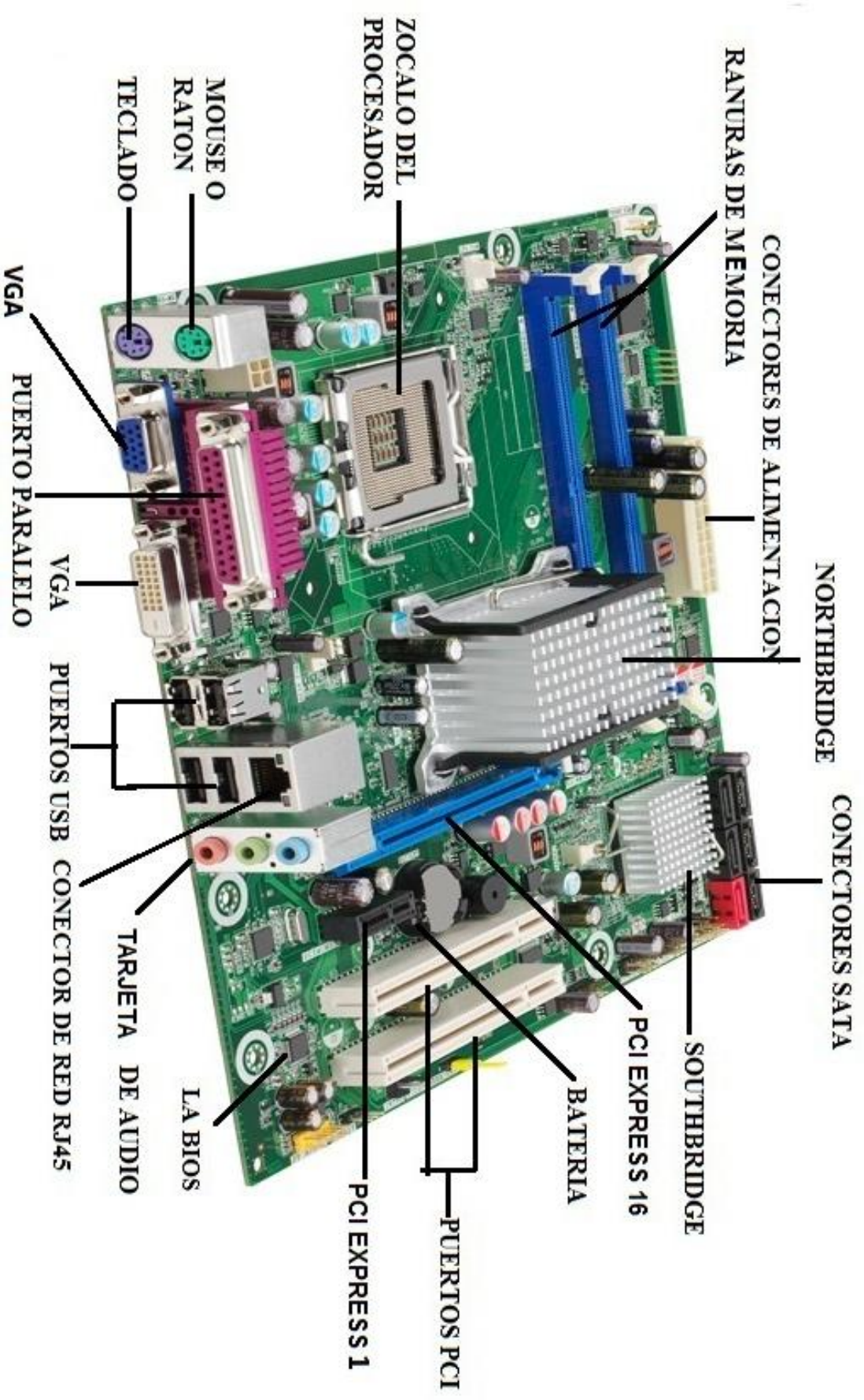
INTRODUCCION TEORICA.

Comenzaremos haciéndonos la siguiente pregunta: ¿Qué es la tarjeta madre?
La tarjeta madre, conocida también como motherboard (en ingles), es el elemento principal de toda computadora, ya que en ella se encuentran o se conectan todos los demás dispositivos necesarios para el correcto funcionamiento de una computadora.

Una tarjeta madre no es más que un circuito impreso en la cual se encuentra toda la electrónica. Existen diferencias en forma de las tarjetas madre, pero todas poseen elementos que son comunes, y que lo que cambian es su ubicación, y hay elementos que son propios de una tarjeta madre, y que posiblemente no encontremos en otra, como por ejemplo una motherboard que maneje dispositivos SCSI tendrá un chip controlador del puerto, mientras que las tarjetas que no puedan manejar directamente este tipo de dispositivos, no encontraremos dicho integrado (controlador). Es importante poder identificar cada uno de estos elementos dentro de la tarjeta madre, así como conocer cual es su función en ésta. Siempre encontraremos en el manual de la motherboard un diagrama detallado de cada una de las partes principales de esta, así como una breve descripción de las funciones de esta y su configuración.
En la siguiente figura observaremos una motherboard en la cual se detallan cada uno de estos elementos principales.



Tarjeta madre y sus partes principales



Tarjeta madre y sus partes principales

A continuación describiremos cada una de las funciones de estas partes de manera general.

Comenzaremos por una de las partes principales, el ZIF Socket.

ZIF Socket.

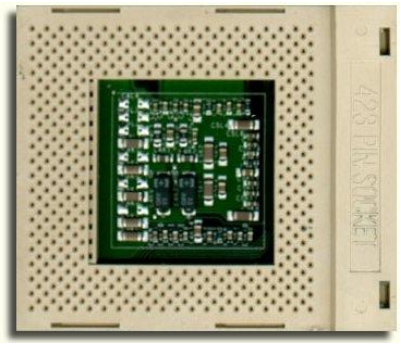
¿Por qué ZIF? El nombre proviene del ingles “Zero Insertion Force” traducido al español como “fuerza cero de inserción”. También llamados Zócalos de los microprocesadores. Un socket es el lugar donde se instala el microprocesador y no son los mismos para todos los procesadores, estos pueden variar en forma, por lo que cada uno tiene un determinado nombre, es mas, con el Pentium II se cambio de la estructura física cuadrada al de una ranura (slot en ingles)

Entre algunos de los tipos de socket del tipo ZIF podemos encontrar:

- Socket 7 y “Super Socket 7”
- Socket A
- Socket 423 (utilizado por los Pentium 4)

En la siguiente figura podemos ver la forma particular de algunos sockets ZIF:

Socket 7

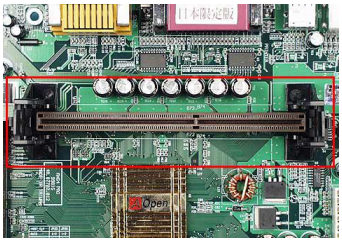


Socket 423

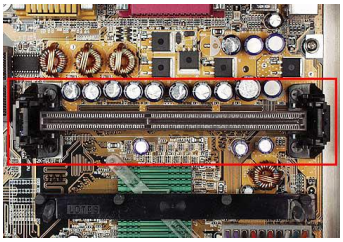
Por otro lado podemos encontrar zócalos del tipo “slot 1”, creado y con todos los derechos de Intel y el “slot A” creado por la AMD. Cada uno corresponde a un diseño propietario por lo que no tomaremos tiempo en

hablar de el, solo agregaremos que entre ellos dos son totalmente incompatibles. Podemos ver en la siguiente figura la total diferencia con el ZIF

Microprocesador



Slot 1



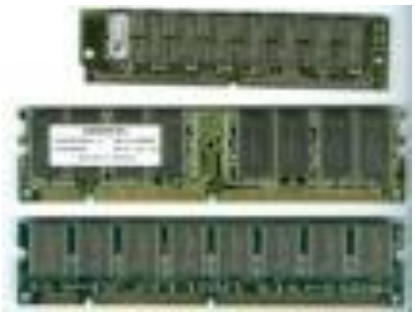
Slot A

Es de agregar que hay tarjetas en las que no se encuentra ningún zócalo, esto es por que el microprocesador esta directamente soldado a la tarjeta, lo que suele muchas veces hacer difícil identificarlo. También hay microprocesadores mas antiguos con una forma rectangular, muy similar al aspecto de la BIOS.

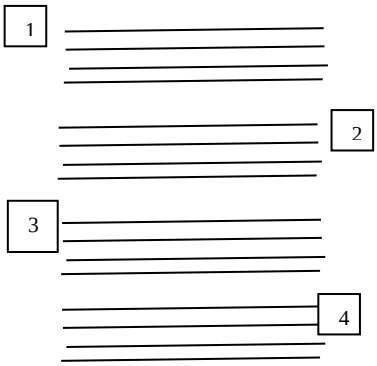
Ranuras de Memoria, y las Memorias RAM

La memoria RAM es el lugar donde se guardan datos que se utilizan en el momento presente., y es un tipo de memoria volátil, es decir que cuando se apaga la computadora se pierde la información guardada en ella.
Existen muchos tipos de memoria RAM, y para cada una de ellas su respectiva ranura, la cual ha ido cambiando a lo largo del tiempo. Se han ido mejorando las velocidades de almacenamiento y toma de datos. En la tarjeta madre de arriba, se tiene un slot o ranura que corresponde a una memoria SDRAM, la cual es un tipo de memoria especifica.

En la siguiente figura podremos observar el aspecto de tres memorias RAM, en las cuales los rectángulos negros (chips) son las memorias, las cuales están soldadas al impreso. Al conjunto total se le llama modulo.



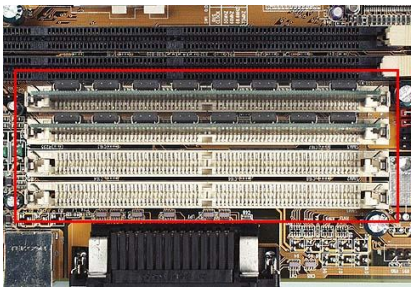
Memorias RAM



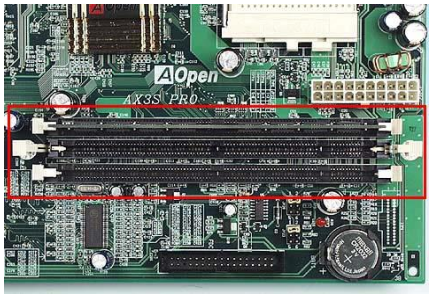
Existen diversos tipos de memoria RAM, entre los que podemos mencionar:

- DRAM: “Dinamic RAM” o RAM Dinámica.
- Fast Page: “Pagina Rápida” Llamada también DRAM, ya que evoluciona de la anterior, y se caracteriza por ser un poco mas rápida.
- EDO: Denominada también EDO-RAM “Extended Data Output-RAM” evoluciona de la Fast Page y tiene la característica de permitir la entrada de datos mientras los anteriores están saliendo.
- SDRAM: “Sincronic-RAM” o RAM Sincronizada. Se le llama así ya que funciona de manera sincronizada con la velocidad de la tarjeta madre.
- PC66, PC100 y PC133: Llamadas SDRAM de 66, 100 y 133 MHz respectivamente, por trabajar a esta frecuencia.
- DDR: “Doble Data Rate” Denominada también como SDRAM II por duplicar la eficiencia del chip de memoria. Esto lo logra ya que puede leer tanto en la parte superior como la inferior del reloj de la tarjeta madre.

Por otro lado, dependiendo de la forma en que se juntan los chips de memoria para poder conectarse estos a la tarjeta madre, podemos distinguir entre dos tipos de módulos: los SIMMs y DIMMs.
Los SIMMs “Single In-Line Memory Module” con 30 ó 72 contactos. El primero puede manejar 8 bits y mide aproximadamente 8.5 cm, mientras que el segundo maneja 32 bits y mide aproximadamente 10.5 cm. Por lo general el zócalo donde se colocan es de color blanco.
El DIMM es mas alargado (por el manejo de mas bits) mide aproximadamente 13 cm. Tiene 168 contactos y se colocan por lo general de color negro. Puede manejar 64 bits de una sola vez, lo que le permite poder usarse de uno en uno en los procesadores Pentium (en SIMMs se manejaban de dos en dos para cubrir los 64 bits).



Ranuras SIMMs



Ranuras DIMMs

El CHIPSET

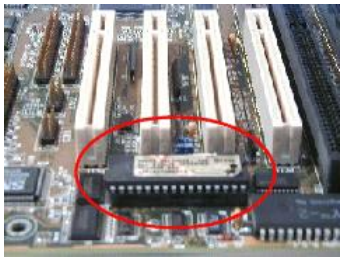
Es un conjunto de Chips, de ahí su nombre. Este es muy importante en el desempeño del computador. Se encarga de controlar determinadas funciones de la computadora, como por ejemplo como interaccionara la memoria cache con el microprocesador, se encarga del manejo de lo puertos de expansión tales como el ISA, PCI, AGP, USB, etc. En otras palabras se encarga de dirigir el flujo de datos de un punto a otro, asegurando que esta información se dirija al lugar apropiado. Cada uno de los chips que se encuentran dentro del chipset tiene una función particular, es decir, cada una de las tareas que mencionamos anteriormente, la desempeña cada uno de estos chips.

El aspecto de un chipset es el siguiente:

Chipset

LA BIOS.

Sus siglas provienen de "**B**asic **I**nput/**O**utput **S**ystem", o lo que es lo mismo "Sistema Básico de Entrada y Salida". Consiste en un chip el cual tiene grabado un programa el cual se encarga de realizar los test post-arranque verificándose el correcto funcionamiento de los componentes de la placa, por ejemplo, se encarga de probar la memoria, video, etc.



BIOS

CMOS

La CMOS es una memoria que se encarga de mantener la información sobre la configuración de la computadora. Es alimentada mediante una pila recargable. Cada vez que se agrega algún dispositivo, es necesario configurar el equipo instalado, y grabar estos cambios en la memoria, de manera que cuando la computadora arranque, pueda identificar cada uno de los componentes que se grabaron en la memoria. En ella se graban también otras configuraciones tales como la fecha y hora, password, secuencias de arranque, etc.

Entre los puntos de importancia a la hora de resolver problemas en el arranque de la computadora es considerar los pitidos que esta da dependiendo del tipo de falla. A continuación presentamos los pitidos que pueden darse y que falla representa:

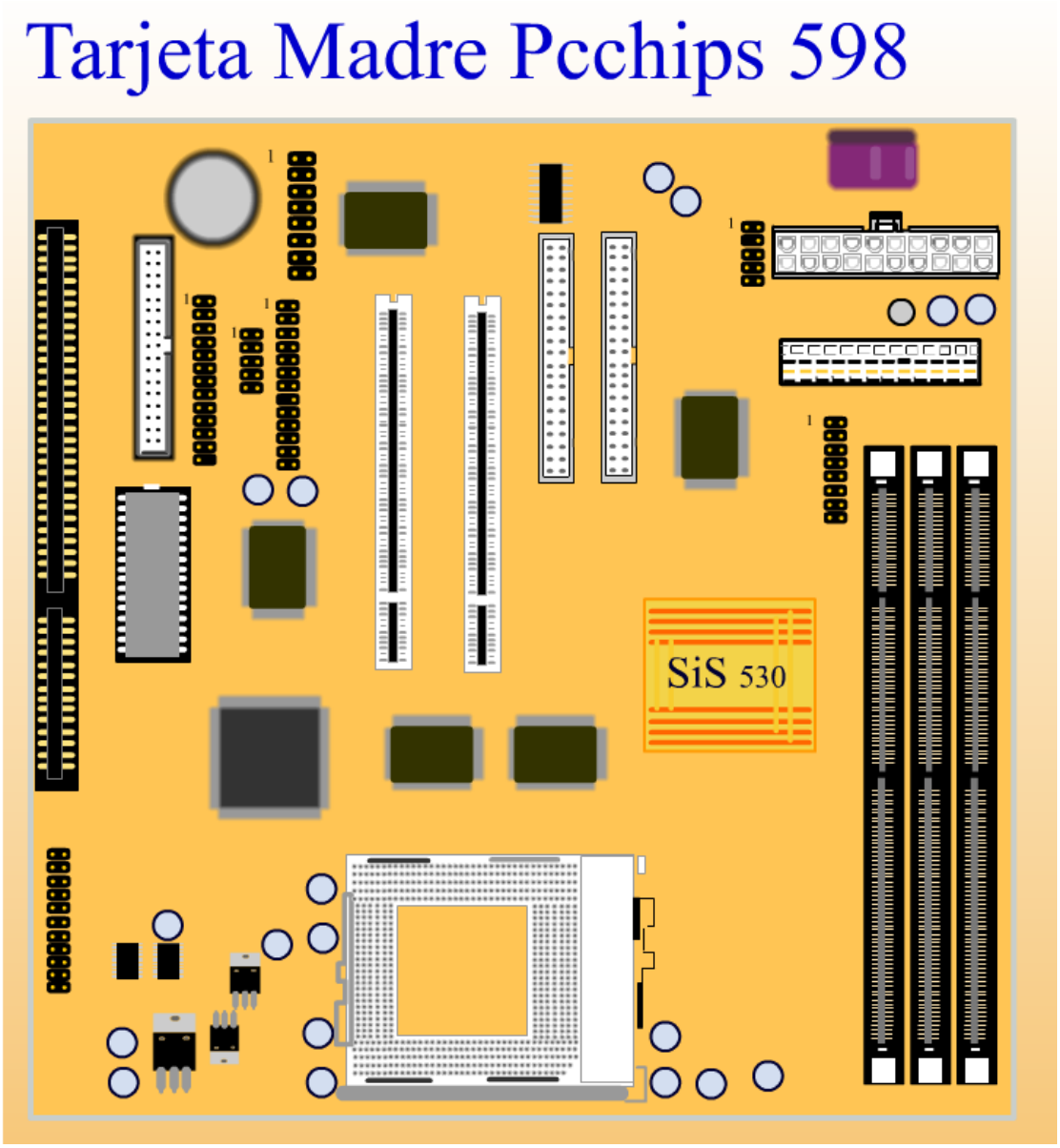
Tabla

No hay pitidos	No hay suministro eléctrico o el parlante de sistema esta desconectado o defectuoso
1 pitido corto	Arranque normal.
Pitido constante ininterrumpido	Falla en el suministro eléctrico
Pitidos cortos y constantes	Tarjeta madre defectuosa.
1 pitido largo	Error de memoria RAM. **RAM Refresh Failure**. Los diferentes componentes encargados del refresco de la memoria RAM fallan o no están presentes.
1 pitido largo, 1 corto	Fallo general en la placa madre o ROM básica del sistema
1 pitido largo,2 cortos	No se encuentra la tarjeta de video, puede estar mal instalada o bien defectuosa.
1 pitido largo, 3 cortos	No se encuentra monitor conectado a la tarjeta gráfica.
1 pitido largo, varios cortos	Falla relacionada con el video (depende del tipo de tarjeta de video y del la configuración de la bios)
2 pitidos largos y uno corto	Falla en la sincronización de imágenes
2 pitidos cortos	Error de paridad de memoria. La paridad no es soportada por la placa base.
3 pitidos cortos	Fallo de memoria en los primeros 64 Kbytes de la RAM.
4 pitidos cortos	El temporizador o contador de la placa base se encuentra defectuoso. El Timer 1 de la placa no funciona.
5 pitidos cortos	La CPU ha generado un error porque el procesador o la memoria de vídeo están bloqueada.
6 pitidos cortos	El controlador o procesador del teclado (8042) puede estar en mal estado. La bios no puede conmutar en modo protegido. Este error se suele dar cuando se conecta/desconecta el teclado con el computador

	encendido.
7 pitidos cortos	La CPU ha generado una interrupción excepcional o el modo virtual del procesador está activo.
8 pitidos cortos	El adaptador de vídeo (tarjeta gráfica) del sistema no existe o su memoria de vídeo (RAM) está fallando. No es un error fatal. Es un fallo de escritura de la Video RAM.
9 pitidos cortos	Error de conteo de la Video RAM. El valor del checksum (conteo de la memoria) de la RAM no coincide con el valor guardado en la bios.
10 pitidos cortos	El registro de la CMOS RAM falla a la hora de la desconexión.
11 pitidos cortos	La memoria caché externa está fallando. En la bios hay una opción llamada video memory cacheable, que lo que hace es volcar el contenido de la ram de la tarjeta en el disco duro. Ponla en enable, haber si soluciona tu problema.
2 pitidos cortos	Se ha detectado un error al realizar uno de los tests de hardware.
Pitidos cortisimos, tenues y constantes	Posiblemente el teclado provoca este error o bien una tecla presionada durante el arranque.

Fuente:
http://enlaces.ucv.cl/faq/Soluciones/Arranque/Detalle_Pitidos.htm

Identificar las partes principales de la siguiente figura



¿Cómo seleccionar el mejor chipset para mi computador?
La forma correcta y sencilla de seleccionar la mejor Chipset para el computador, es determinar el tipo de equipo que vamos a montar, por ejemplo, en un procesador económico tipo Pentium, Celeron o Core i3, la mejor opción sería el H110.
Por otro lado, si pensamos en utilizar Intel Optane, sin el procesador serie K, la mejor opción sería H270, ya que cubre con las necesidades y expectativas que posee cualquier usuario que no vaya a utilizar configuraciones gráficas multiGPU y soluciones preparadas de overclock.
Por último, se encuentran Z270, Z170 y Z370, ideales para un procesador serie K de Intel, ya que son los únicos modelos que permiten hacer overclock

modelos de Chipset que hay en el mercado

- Chipset Intel H370
- Intel H110
- Chipset Intel B360
- Intel B365
- Chipset Intel Z370
- Intel Z390
- Chipset Intel X79 Express
- Intel Z68 Express
- Chipset Intel H55 Express
- Intel H310
- Chipset AMD X370
- AMD A320
- Chipset AMD B350

Dada las arquitecturas y configuraciones de la tarjeta madre (Motherboard), esta nos da una característica para poder realizar un RAID de disco, tomemos en cuenta que no todas las tarjetas madres para máquinas de escritorio soportan arreglos de discos, en nuestro caso las maquinas DELL VOSTRO que posee en el laboratorio nos permite realizar arreglos de disco.

Pedir a su instructor que explique y realizar la práctica para su configuración, dependiendo de hasta que RAID soporta nuestro equipo y la cantidad de discos duros que se tiene para realizarla.

Los registros de la CPU

El programa se puede llamar debug.com o debug.exe, dependiendo de la versión de DOS que tenga usted. El Debugger puede ser invocado de dos formas: la primera se reduce a digitar en la línea para comandos debug seguido de la tecla [Enter], y la segunda consiste en proporcionar el nombre del archivo sobre el cual deseamos operar, digitando debug de archivo y posteriormente la tecla [Enter]. Iniciemos con la primera opción; digite:

C:\Debug [Enter]

Verá que aparece un guión en la siguiente línea. Este guión le indica que Debug se encuentra esperando algún comando.
Digite:

- r [Enter]

Donde r es el comando REGISTER del Debug y su propósito es exhibir o modificar uno o más registros de la CPU. La sintaxis de este comando es r [REGISTRO], donde [REGISTRO] es la notación de un registro válido de la CPU y consta de dos caracteres alfabéticos siguiendo la nomenclatura de los microprocesadores 808x y 80x86.

Las combinaciones de caracteres aceptables son:

AX, BX, CX, DX, DS, ES, CS, SS, SI, DI, IP, SP, BP y F

Si no se indica el nombre de un registro, Debug desplegará el contenido de todos ellos. Si se indica el nombre del registro, el contenido de éste será desplegado en notación hexadecimal y en la línea de abajo aparecerán dos puntos [:].

En este momento se tienen dos opciones: cambiar el valor del registro digitando el nuevo valor, o digitar r para dejar el valor original. Veamos algunos ejemplos:

Después de haber digitado el comando "**r**" y [Enter], como resultado se desplegará lo siguiente:

```
AX=0000 BX=0000 CX=0000 DX=0000 SP=FFEE BP=0000 31=0000 DI=0000
DS=24E4 ES=24E4 SS=24E4 CS=24E4 IP=0100 NV UP El PL NZ NA PO NC
24E4:010007     POP ES
```

El contenido de los registros que se desplieguen en su pantalla quizá sean diferentes, pero la estructura es la misma.

Véase la Figura 1.9. La primera línea muestra lo que contienen algunos registros de la CPU.

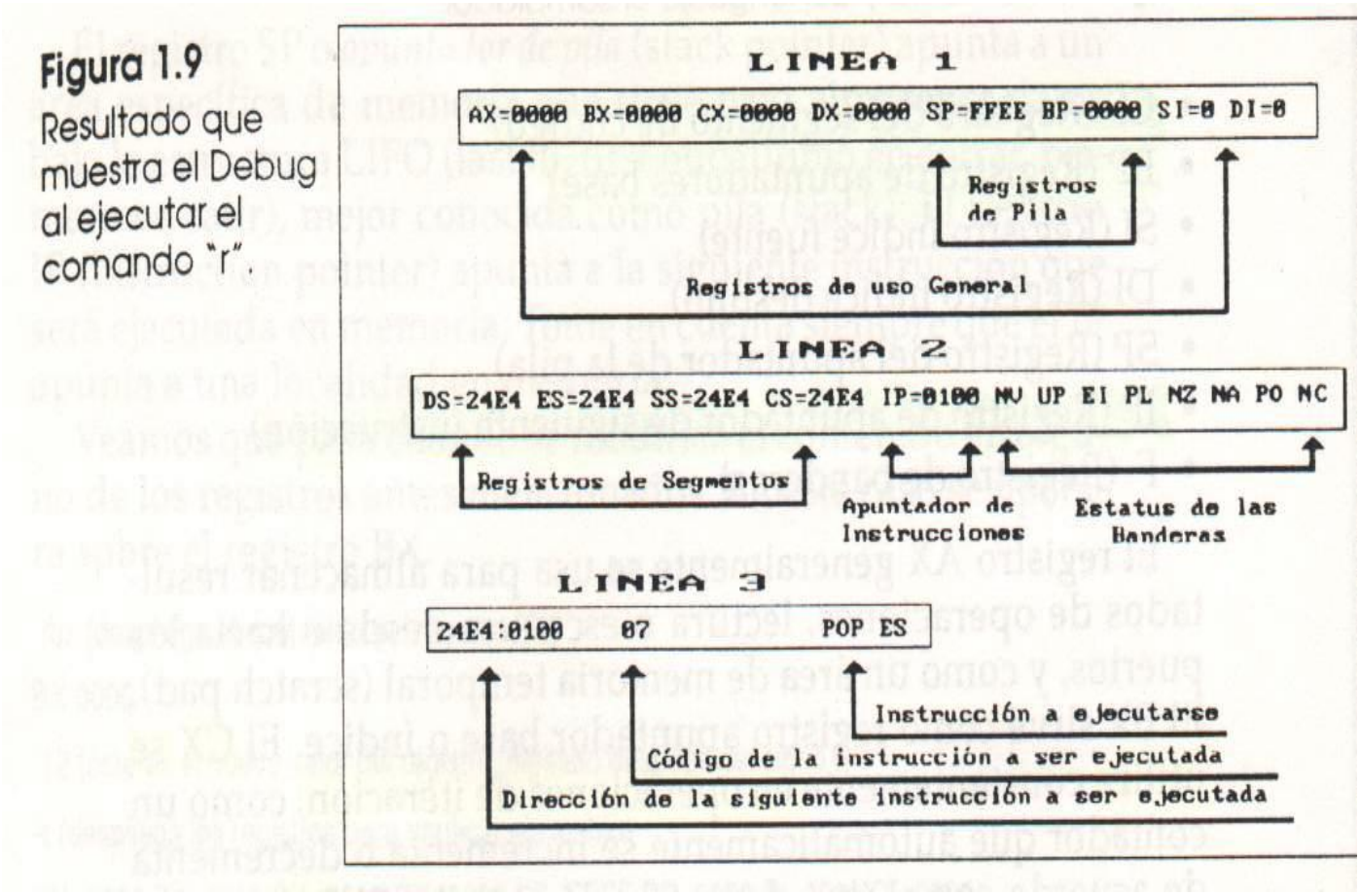


Figura 1.9
Resultado que muestra el Debug al ejecutar el comando "r".

La segunda línea continúa mostrando el contenido de los registros restantes, así como el estado actual del registro de banderas. Finalmente, la tercera línea muestra cuál es la instrucción que está a punto de ser ejecutada. Observe que después de CS=24E4 tenemos el contenido del registro IP. En este caso es 100H, y es en esta dirección donde se encuentra la instrucción POP ES. La cuarta línea contiene un guión, el cual indica que Debug se encuentra esperando el siguiente comando. Para alterar el valor de algún registro basta con digitar el comando **r** seguido por el nombre del registro. Sin embargo, antes de detallar lo anterior es necesario describir los registros de la CPU, así como sus características más importantes.

La CPU tiene 14 registros internos, cada uno de 16 bits. Los primeros cuatro, AX, BX, CX y DX son registros de uso general y se pueden usar también como registros de 8 bits. Es decir, AX se puede dividir en AH y AL (AH es el byte alto -High- y AL es el byte bajo -Low-). Lo mismo es aplicable a los otros
tres (BX en BH y BL, CX en CH y CL, y DX en DH y DL).

Estos son los únicos registros que pueden usarse de modo, dual (en 8 o 16 bits). En la práctica. Los registros de la CPU generalmente son conocidos por sus nombres propios, los cuales se presentan en seguida:

- AX (acumulador)
- BX (Registro base)
- CX (Registro contador)
- DX (Registro de datos)
- DS (Registro del segmento de datos)

• ES (Registro del segmento extra)
• SS (Registro del segmento de pila)
• CS (Registro del segmento de código)
• BP (Registro de apuntadores base)
• SI (Registro índice fuente)
• DI (Registro índice destino)
• SP (Registro del apuntador de la pila)
• IP (Registro de apuntador de siguiente instrucción)
• F (Registro de banderas)

El registro AX generalmente se usa para almacenar resultados de operaciones, lectura o escritura desde o hacia los puertos, y como un área de memoria temporal (scratch pad).

El BX sirve como registro apuntador base o índice. El CX se utiliza constantemente en operaciones de iteración, como un contador que automáticamente se incrementa o decrementa de acuerdo con el tipo de instrucción usada. El DX comúnmente se usa como puente para el acceso de datos.
El DS es un registro de segmento cuya función es actuar como policía donde se encuentran los datos. Cualquier dato, ya sea una variable inicializada o no, debe encontrarse dentro de este segmento. La única excepción es cuando tenemos programas del tipo .COM, ya que en éstos sólo puede existir un segmento, como se verá más adelante. El registro ES tiene el propósito general de permitir operaciones sobre cadenas, pero también puede ser una extensión del DS.

El SS tiene la tarea exclusiva de manejar la posición de memoria donde se encuentra la pila (stack). Esta es una estructura usada comúnmente para almacenar datos en forma temporal, tanto de un programa como de las operaciones internas de la PC. En términos de operación interna, la CPU usa este segmento para almacenar las direcciones de retorno de las llamadas a rutinas. El registro de segmentos más importante es el CS o segmento de código. Es aquí donde se encuentra el código ejecutable de cada programa, el cual está directamente ligado a los diferentes modelos de memoria.

El registro BP (base pointer) se usa para manipular la pila sin afectar al registro de segmentos SS. Generalmente es muy útil cuando se usa la interfaz entre lenguajes de alto nivel y el ensamblador. Puesto que dicha interfaz se basa en el concepto de la pila BP, permite accesar parámetros pasados sin alterar el registro de segmento SS. Los registros SI y DI son útiles para manejar bloques de cadenas en memoria, siendo el primero el índice fuente y el segundo el índice destino. En otras palabras, SI representa la dirección donde se encuentra la cadena, y DI la dirección donde será copiada.

El registro SP o apuntador de pila (stack pointer) apunta a un área específica de memoria que sirve para almacenar datos bajo la estructura LIFO (last in, first out: último en entrar, primero en salir), mejor conocida como pila (stack). El registro IP (instruction pointer) apunta a la siguiente instrucción que será ejecutada en memoria. Tome en cuenta siempre que el IP apunta a una localidad en memoria.

Veamos qué pasa cuando se modifica el contenido en alguno de los registros antes mencionados. En este caso se operará sobre el registro BX.

- rbx (despliega el contenido del registro BX)

BX 0000

:12 (éste es el nuevo valor del registro, digitado después del signo:)

-r (despliega los registros para verificar el cambio)

```
AX=0000 BX=0012 CX=0000 DX=0000 SP=FFEE BP=0000 SI=0000 Dl=0000
DS=24E4 ES=24E4 SS=24E4 CS=24E4 IP=0100 NV UP EI PL NZ NA PO NC
24E4:010007     POP ES
```

Se comprueba que el cambio fue efectuado.
También podemos alterar el valor del registro de banderas (véanse los bits del registro en la Figura 1.9). Dicho registro es la base para efectuar bifurcaciones basadas en el resultado de alguna operación.

Si el resultado de comparar dos registros es cero, entonces, NZ sería ZR (lo cual significa que el resultado de la operación es cero). A continuación se describe el significado de cada bit del registro de banderas.

Todas las banderas apagadas :
NV UP DI PL NZ NA PO NC
Todas las banderas prendidas:
OV DN El NG ZR AC PE CY
Significado de los bits:

- Overflow NV=no hay desbordamiento; OV=sí lo hay
- Direction UP=hacia adelante; DN=hacia atrás
- Interrupts DI=desactivadas; EI=activadas
- Sign PL=positivo; NG=negativo
- Zero NZ=no es cero; ZR=sí es cero
- Auxiliary Carry NA=no hay acarreo aux. AC=hay acarreo aux.
- Parity P0=paridad non PE=paridad par
- Carry NC=no hay acarreo CY= sí lo hay

El registro de banderas es un registro de 16 bits, pero como habrá notado no todos los bits se usan.

Interrupciones

De alguna manera, la CPU tiene que estar consciente de lo que sucede a su alrededor. Esta "conciencia" la adquiere mediante las interrupciones. Cada vez que se oprime una tecla, el teclado interrumpe a la CPU y esta detiene lo que está haciendo y pone atención al teclado. Obtiene el código de la tecla presionada y lo guarda en una parte de su memoria llamada memoria intermedia de teclado (type ahead buffer). Otra interrupción que ocurre automáticamente, aproximadamente 18 veces por segundo, es la que actualiza la hora del día. Esta actualización NO se basa en un reloj de tiempo real sino en el reloj que mantiene la PC al iniciar sus operaciones.
Las Interrupciones ocurren tan seguido que la CPU necesita un modo eficiente de manejarlas. Para ello usa un circuito integrado (chip) que le permite saber cuál es la prioridad de cada interrupción (a nivel de hardware). Una manera de administrar las interrupciones, que se usa en el entorno MS-DOS, es mantener una tabla de vectores en la memoria baja, empezando por la localidad 0 y terminando en la 256 (decimal).
Estos vectores de interrupción señalan otra localidad de memoria donde la CPU empezará a ejecutar el código que ahí se encuentre. A esta técnica se le llama darle servicio a la interrupción.
En pocas palabras, una interrupción es una bifurcación a cierta localidad de memoria (RAM o ROM) donde la CPU iniciará la ejecución de una serie de instrucciones, y al terminar regresará a la siguiente localidad de la instrucción que causó la interrupción.

Las interrupciones se dividen en:

1. Interrupciones de hardware
2. Interrupciones de software

Las interrupciones de software se dividen a su vez en:

1. Interrupciones de BIOS
2. Interrupciones de DOS

Las interrupciones de BIOS (Basic Input/Output System: sistema básico de entrada/salida) son aquéllas que, en su mayoría, se encargan de controlar los periféricos/ aunque en algunas ocasi

Las interrupciones de DOS (Disk Operating System) son las que se encargan de administrar tanto la memoria como el disco. También existen interrupciones que manejan la entrada y salida de información.
Las interrupciones de DOS generalmente se usan cargando el parámetro de la función deseada en el registro AH e invocando la interrupción 21H. A este proceso se le llama utilizar las funciones de DOS. En la práctica, esta interrupción resulta ser la más usada. Una razón para ello es la necesidad de mantener la compatibilidad a través de las futuras versiones del sistema operativo MS-DOS.

La reacción de la PC ante una interrupción

LA CPU se encuentra realizando alguna tarea, y de pronto el teclado interrumpe. De inmediato brinda el servicio al teclado, viendo primero cuál es la dirección asignada a la interrupción en la memoria baja. Los vectores de memoria baja se encuentran numerados del 0 al 256 (dos bytes corresponden al segmento y otros dos al desplazamiento). Es decir, por cada interrupción se mantienen 4 bytes en memoria. Conforme al número de la interrupción, "levanta" la dirección especificada y salta a ella, ejecuta el código a partir de dicha localidad, y regresa para continuar con lo que estaba haciendo a la localidad siguiente de la que causó la interrupción.
Ejemplo;
Veamos algunos ejemplos:
Programa:

```
C:\Debug
-al00
2343:0100   mov  ah, 1   ; función para cambiar el cursor
2343:0102   mov  cx, 7   ; forma del cursor
2343:0105   int  10      ; invoca al BIOS
2343;0107   int  20      ; termina el programa
2343:0109
```

El programa anterior utiliza la función 1 de la interrupción de BIOS, la cual cambia la forma del cursor. Esta interrupción exige que la función a realizar sea puesta en el registro AH, y el código del cursor en CX. El registro CX divide el código mencionado en:

CH = Línea inicial del cursor (bits 4-0)
CL = Línea final del cursor (bits 4-0)

Para regresar el cursor a su modalidad predefinida (que es una subraya) basta con cambiar la línea que dice 'mov cx,7' por 'mov cx,0607'. Esto es válido para las tarjetas adaptadoras que son compatibles con color.

Veamos un ejemplo usando las interrupciones de DOS:

Programa:

```
-a100
2343:   0100    mov     ah,1    ;función 1      (lee el teclado)
2343:   0102    int     21      ;invoca al      DOS
2343:   0104    cmp     al,0d   ;compara si lo leído es un retorno del carro)
2243:   0106    jne     100     ;si no lo es, lee otro carácter
2343    0108    mov     ah,2    ;función 2      (escribe en la pantalla)
2343    010A    mov     dl,al   ;carácter       a escribir en al (es un)
2343    010C    int     21      ;Invoca al      DOS
2343    010E    int     20      ;regresa el control a Debug
2343    0110
-g
abcdef
Program terminated normally (Programa terminado normalmente)
```

El ejemplo anterior utiliza la interrupción 21H del DOS. Emplea dos funciones de la misma: la primera lee el teclado (función 1) y la segunda escribe en la pantalla. Este programa lee caracteres del teclado hasta que encuentra un <CR>.
Como punto final a esta sección correspondiente al Debuger, veamos un programa que realiza algo práctico, usando tanto las interrupciones de BIOS como de DOS. El programa que a continuación se presenta permite ocultar o mostrar un archivo. Después de ocultar el archivo no lo podrá ver al obtener el directorio. Puede usarlo como base de un sistema para proteger información.

El programa no contiene todas las facilidades de verificación que se incluirían normalmente, ni tampoco otras funciones; eso se deja a su imaginación y creatividad. Algunas de las adiciones que puede hacerle son:

- Comprobar que la letra (E) o (M) sea válida (E=oculta y M=muestra). No permitir letras que no sean las anteriores.
- Verificar que el archivo exista en disco.
- Verificar que en el nombre del archivo se digite la extensión.
- Permitir que el programa sea invocado desde la línea para comandos con los parámetros de función y el nombre del archivo.

ej. ESMUES E X.ASM

Nota: Observe que al invocar al programa desde la línea para comandos, debemos separar los parámetros mediante espacios(E) y (X.ASM). Primero se describirá el algoritmo y después el programa:

```
INICIA:
PONER EN BLANCO LA PANTALLA;
PREGUNTAR POR LA FUNCIÓN A REALIZAR;
PREGUNTAR POR EL NOMBRE DEL ARCHIVO;
IF FUNCIÓN ES = 'E' THEN
     OCULTAR EL ARCHIVO
ELSE
    MOSTRAR EL ARCHIVO
ENDIF
TERMINA
```

Veamos por partes el algoritmo:
Poner en blanco la pantalla
Existen varias maneras de poner en blanco la pantalla. La elegida no es de uso muy común, pues se basa en el funcionamiento del hardware y en las modalidades de video. Se usa la interrupción 10 del BIOS con la función "cambiar modalidad de video" (cambia a 80 X 25 en blanco y negro). Toma tres instrucciones.

Pregunta la función a realizar

Existen varias maneras de desplegar caracteres en la pantalla. Una es desplegar carácter por carácter, y la otra es desplegar una cadena de caracteres. Se eligió la segunda por su sencillez y porque en realidad estamos desplegando una cadena de caracteres. Se usa la interrupción 21 de DOS en su función 9. Esta dice que el registro par DS:DX requiere la dirección donde se encuentra el primer carácter de la cadena en memoria. Así que se carga el registro DX con dicha dirección. La función a realizar debe ser pasada en el registro AH (la función es 9).

Una vez desplegada la cadena, de alguna manera se debe obtener la respuesta. Aquí se tienen de nuevo dos opciones.

La primera es desplegar un carácter a la vez, y la segunda es desplegar una cadena que termine con un <CR>. Se eligió la primera porque sólo es necesario un carácter (E o M). La respuesta se obtiene mediante la interrupción 16H (interrupción del teclado) del BIOS. Esta exige que el registro AH contenga un cero (que es el número de la función). La interrupción 16 regresa en el registro AL el valor ASCII de la tecla, o un 0 (el cero es regresado en caso de que la tecla sea alguna diferente a la tabla ASCII; pueden ser combinaciones de o alt, etc.). El registro AH regresará el código extendido de la tecla si es que AL=0.

Pregunta el nombre del archivo

Igual a la anterior, con la excepción de cómo se maneja la respuesta. Puesto que la respuesta es el nombre del archivo y ésta es una cadena de caracteres, se optó por usar una función de la interrupción 21H del DOS. Esta es la función OAH y exige que se le pase en el registro DS:DX un apuntador a la dirección de la memoria intermedia (buffer) que se usará para

guardar el resultado en memoria. El contenido de la memoria intermedia debe estar inicializado de la siguiente
forma:

- El primer byte debe contener el número total de caracteres a leer, incluyendo el.
- El segundo byte contendrá la cantidad actual de caracteres leídos (este byte es inicializado por la función, no necesita hacerlo usted).
- Lo que dígito será puesto en la memoria intermedia (buffer) a partir del tercer byte.

IF FUNCIÓN ='E' ENTONCES OCULTA EL ARCHIVO
ELSE
MUESTRA EL ARCHIVO
ENDIF

Cuando se digitó la función a realizar, ésta fue guardada en una localidad de memoria para poder recuperarla posteriormente. Si no se hiciera así, podría suceder que la llamada a alguna otra función destruyera el contenido del registro que tenía el resultado, provocando que el programa no sepa qué hacer. Una vez definida la función a realizar se invoca la función 43H de la interrupción 21H (Get/Set file attribute). A continuación se detallan los pasos que seguirá esta función:

- El registro AH debe tener la función a realizar (en este caso 43H).
- El registro AL debe tener un 1 (uno), indicando que se pondrán los atributos del archivo.
- El registro CX deberá contener el atributo a usar de la siguiente lista:

00 = normal (puede ser leído o escrito sin restricciones)
01 = Sólo lectura
02 = Oculto
04 = De sistema
08 = volumen (sólo un archivo puede tener este atributo, y debe estar en el directorio raíz).
10 = Subdirectorio

Nota: Todos los valores de los atributos anteriores están en hexadecimal.
El registro DX deberá tener el apuntador o la dirección del nombre del archivo en el formato de cadena ASCIIZ. Una cadena ASCIIZ es una cadena de caracteres que al final tiene un carácter nulo. Es decir, una secuencia de 00.

Programa:

c:\debug

```
-a100
27BD:0100    jmp 138
27BD:0102
-e 102 'Función a realizar (E) o (M) : $'    ; primera pregunta
-e 122 'Nombre del Archivo ; S'              ; segunda pregunta
-e 250 d 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0     ; el buffer
-e 246 0d 0a '$'                   ; CR y LF.
-a138
27BD:  0138 mov ah,0          ;cambia la modalidad de video a 80
27BD:  013A mov al,2          ;X 25 (blanco y negro) para po-
27BD:  013C int 10            ;ner en blanco la pantalla.
27ED:  013E mov dx,102        ;prepara para desplegar la primera
27BD:  0141 mov ah,9          ;pregunta, mediante la función 9
27BD:  0143 int 21            ;de la interrupción 21.
27BD:  0145 mov ah, 0         ;Obtiene un carácter del teclado
27BD:  0147 int 16            ;mediante la interrupción 16
27ED:  0149 mov [249],al      ;guarda el resultado en la localidad
```

```
27BD:  014C mov dx,246      ;249. Prepara para mandar la se-
27BD:  014F mov ah, 9       ;cuencia <CR> y <LF> a pantalla.
27BD:  0151 int 21          ;¡Mandala'
27BD:  0353 mov dx, 122     ;Prepara para desplegar la segunda
27BD:  0156 mov ah,9        ;pregunta.
27BD:  0158 int 21          ;Despliega la pregunta
27BD:  015A mov ah, 0a      ;Obtiene la cadena de caracteres de respuesta
27BD:  0l5C mov dx,250      ;dirección del buffer (donde guarda-
27BD:  015F int 21          ;remos el nombre del archivo).
27BD:  0161 mov al,[249]    ;restaura la función a realizar
27BD:  0164 cmp al,45       ;¿es E? (oculta archivo)
27BD:  0166 jnz 16D         ;Salta si no lo es
27BD:  0168 mov cx,2        ;Atributo para ocultar archivo
27BD:  016B jmp 170         ;salta a la ejecución de la función
27SD:  016D mov cx, 0       ;atributo que permite mostrar el archivo
27D2:  0170 mov di,252      ;busca el final del nombre. Esto lo
27D2:  0173 mov al,0d       ;sabremos al encontrar el <CR>.
27D2:  0175 scasb           ;en el buffer.
27D2:  0176 jnz 175         ;itera hasta encontrar la posición.
27D2:  0178 move byte ptr[di-01],00 ; Convierte a ASCIIZ
27D2:  017C mov ah, 43      ;Función para cambiar atributos
27D2:  017E mov dx,252      ;dirección del buffer.
27D2:  0181 mov al, 01      ;Comando de cambiar atributo
27D2:  183 Int 21           ;cámbialo
27D2:  0185 int 20          ;Final feliz por Fin

- N esmues.com              ;nombre del programa
- rcx
: 162                       ;Total de bytes que ocupa
- w                         ;escríbalo en el disco
```

En el programa anterior vimos la manera de usar tanto las interrupciones del BIOS como las del DOS. Siga la lógica, es clara y concisa.

Ejercicios para laboratorio

Programas en DEBUG

Ver el contenido de las siguiente instrucción

```
-d 100 12f
Ingresar en el debug la siguiente instrucción
-f 100 12f 'BUFFER'
```

Ver el contenido.

Digitar los siguientes programas

```
-a 100
 xxxx:0100 jmp 126
 xxxx:0102 db 0d,0a,"Este es mi primer programa con DEBUG!"
 xxxx:0123 db 0d,0a,"$"
 xxxx:0126 mov ah,9
 xxxx:0128 mov dx,102
 xxxx:012B int 21
 xxxx:012D mov ah,0
 xxxx:012F int 21       ; programa terminado.
 xxxx:0131
 -g =100
```

Arc-115 (Registros)

Limpiar Pantalla bajo DOS

```
-A 100
     6897:0100 mov ax,600
     6897:0103 mov cx,0
     6897:0106 mov dx,184f
     6897:0109 mov bh,07
     6897:010B int 10
     6897:010D int 20
     6897:010F
```

3

```
N COLDBOOT.COM
A 100
MOV AX,0040
MOV DS,AX
MOV WORD PTR [0072],FFFF
CLI
JMP F000:FFF0

R CX
11
W
Q
```

4

```
N WARMBOOT.COM
A 100
MOV AX,0040
MOV DS,AX
MOV WORD PTR [0072],1234
CLI
JMP F000:FFF0

R CX
11
W
Q
```

ADD (01h,d8) y SUB(29h,d8)
MUL (F7h, E3h) y DIV (f7h, F3h)

Arc-115 (Registros)

Imprimir abecedario

```
-A100
MOV AH,02
MOV DL,60
ADD DL,01
CMP DL,7A
JA 110
INT 21
LOOP 104
ADD DL,07
INT 21
ADD DL,1D
MOV CX,04
ADD DL,01
INT 21
LOOP 11B
INT 20
```

Valores en Hexadecimal

```
A100
0100 MOV AH,02
0102 MOV DL, 2F
0104 ADD DL,01
0107 INT 21
0109 CMP DL,39
010C JL 104
010E ADD DL,07
0111 MOV CX,06
0114 ADD DL,01
0117 INT 21
0119 LOOP 114
011B INT 20
011D
```

APENDICE 1- INTERRUPCIONES.

DIRECCION (hex)	INTERRUPCION (hex)	FUNCION
0-3	0	Division by zero
4-7	1	Single step trace
8-B	2	Non maskable interrupt
C-F	3	Break point instruction
10-13	4	Overflow
14-17	5	Print screen
18-1F	6,7	Reserved
20-23	8	Timer
24-27	9	Keyboard interrupt
28-37	A,B,C,D	Reserved
38-3B	E	Diskette interrupt
3C-3F	F	Reserved
40-43	10	Video screen I/O
44-47	11	Equipment check
48-4B	12	Memory size check
4C-4F	13	Diskette I/O
50-53	14	Communication I/O
54-57	15	Cassette I/O
58-5B	16	Keyboard input
5C-5F	17	Printer Output
60-63	18	ROM Basic entry code
64-67	19	Bootstrap loader
68-6B	1A	Time of day
6C-6F	1B	Get control on keyboard break
70-73	1C	Get control on timer interrupt
74-77	1D	Pointer to video initialization table
78-7B	1E	Pointer to diskette parameter table
7C-7F	1F	Pointer to table for graphics characters
ASCII 128-255		
80-83	20	DOS program terminate
84-87	21	DOS function call
88-8B	22	DOS terminate address
90-93	24	DOS fatal error vector
94-97	25	DOS absolute disk read
98-9B	26	DOS absolute disk write
9C-9F	27	DOS terminate, fix in storage
A0-FF	28-3F	Reserved for DOS
100-1FF	40-7F	Not used
200-217	80-85	Reserved by BASIC
218-3C3	86-F0	Used by BASIC interpreter
3C4-3FF	F1-FF	Not Used

JUEGO DE INSTRUCCIONES DEL 8086.

En la siguiente lista de instrucciones, para la descripción y su sintaxis se recurre a las siguientes abreviaturas:

acum	uno de los acumuladores: AX o AL.
reg	cualquiera de los registros
segreg	uno de los registros de segmento
r/m	uno de los operandos generales: registro, memoria, basado, indexado o basado-indexado
inmed	constante o símbolo de 8 o 16 bits
mem	un operando de memoria: símbolo, etiqueta, variable.
etiqueta	etiqueta de instrucciones.
src	fuente en operaciones de cadena
dest	destino en operaciones de cadena.

Para el 8086

AAA Ajuste ASCII para adición.
AAD Ajuste ASCII para división.
AAM Ajuste ASCII para multiplicación.
AAS Ajuste ASCII para división.
ADC acum, inmed Suma con acarreo.
r/m,inmed
r/m, reg
reg, r/m
ADD acum,inmed Suma.
r/m,inmed
r/m, reg
reg, r/m
AND acum,inmed Operación AND a nivel bit.
r/m,inmed

CALL etiqueta Llamado.
r/m
CBW Convierte byte apalabra.
CLC Limpia bandera de acarreo.
CLD Limpia bandera de dirección.
CLI Limpia bandera de interrupción.
CMC Complementa bandera de acarreo.
CMP acum,inmed Comparación
r/m,inmed
r/m, reg
reg, r/m

CMPS src,dest Comparación de cadenas.
CMPSB Compara cadenas byte por byte.
CMPSW Compara cadenas palabra por palabra.
CWD Convierte palabra a palabra doble.

DAA Ajuste decimal para adición.
DAS Ajuste decimal para substracción.
DEC r/m Decremento.
reg
DIV r/m División.
ESC inmed, r/m Escape con 6 bits.
HLT Alto.
IDIV r/m División entera.
IMUL r/m Mutiplicación entera.
IN accum,inmed Entrada desde puerto.
acum, **DX**
INC r/m Incremento.
reg

INT3 Interrupción3 codificada como un byte.
INT inmed Interrupción0-255.

INTO Interrupción en *overflow*.
IRET Retorno de interrupción.
JMP etiqueta Brinco incondicional.
r/m

J (condición)etiqueta Brinca de acuerdo a las condiciones: **A** (arriba), **AE** (arriba o igual), **B**(siguiente), **BE** (siguiente o igual), **C**(acarreo), **CXZ** (**CX** en cero), **E** (igual),**G** (mayor), **GE** (mayor o igual), **L** (menor), **LE** (menor o igual), **NA** (noanterior), **NAE** (no anterior o igual), **NB** (no siguiente), **NBE**(no siguiente oigual), **NC** (no acarreo), **NE** (no igual), **NG**(no mayor), **NGE** (no mayor oigual), **NL** (no menor), **NLE** (no menor o igual), **NO**(no sobreflujo), **NP** (noparidad), **NS** (no signo), **NZ** (no cero), **O**(sobreflujo), **P** (paridad),**PE**(paridad par), **PO** (paridad impar), **S** (signo), **Z** (cero).
LAHF Carga **AH** con las banderas.
LDS r/m Carga **DS**.
LEA r/m Carga la dirección.
LES r/m Carga **ES**.
LOCK Cierra bus.
LODS src Carga cadena.

LODSB Carga byte de cadena en **AL**.
LODSW Carga palabra de la cadena en **AX**.
LOOP etiqueta Ciclo.
LOOPE etiqueta Ciclo mientras igual.
LOOPNE etiqueta Ciclo mientras no igual.
LOOPNZ etiqueta Ciclo mientras no cero.
LOOPZ etiqueta Ciclo mientras cero.
MOV acum,mem Mueve un valor del segundo al primer operando
r/m,inmed
mem, acum
r/m, reg
r/m,segreg
reg, inmed
reg,r/m
segreg,r/m
MOVS dest, src Mueve cadena.

MOVSB Mueve cadena byte por byte.
MOVSW Mueve cadena palabra por palabra.
MUL r/m Multiplicación.
NEG r/m Niega(complemento a 2).
NOP Operación ociosa.
NOT r/m Invierte valores de bits (complemento a 1).

OR acum, inmed Operación OR a nivel de bit.
r/m,inmed
r/m, reg
reg,r/m
OUTDX, accum Salida por el puerto dado por el primer operando. inmed, acum (inmediato de 8 bits)
POP r/m Recupera valor de la pila.
reg
segreg

POPF Recupera banderas.
PUSH r/m Guarda valor en la pila.
reg
segreg
PUSHF Guarda banderas.
RCL r/m,**1** Rotación a la izquierda con acarreo.
r/m,**CL**
RCR r/m, **1** Rotación a la derecha con acarreo.
r/m, **CL**
REP Repite.
REPE Repite si igual.
REPNE Repite si no igual.
REPNZ Repite si no cero.
REPZ Repite si cero.
RET **[**inmed**]** Regresa después de recuperar bytes de la pila.
ROL r/m,**1** Rotación a la izquierda. r/m, **CL**

ROR r/m,**1** Rotación a la derecha.
r/m, **CL**
SAHF Carga banderas con el valor de **AH**.
SAL r/m, **1** Desplazamiento aritmético a la izquierda.
r/m, **CL**
SAR r/m, **1** Desplazamiento aritmético a la derecha.
r/m, **CL**
SBB acum, inmed Substracción con acarreo.
r/m,inmed
r/m, reg
reg,r/m
SCAS dest Explora cadena.
SCASB Explora cadena para el byte en **AL**.
SCASW Explora cadena por la palabra en **AX**.
SHL r/m, **1** Desplazamiento a la izquierda.
r/m, **CL**
SHR r/m, **1** Desplazamiento a la derecha.

r/m, **CL**
STC Fija bandera de acarreo.

STD Fija bandera de dirección.
STI Fija bandera de interrupción.
STOS dest Guarda cadena.
STOSB Guarda byte en **AL** en la cadena.
STOSW Guarda palabra en **AX** en la cadena.

SUB accum, inmed Substracción.
r/m,inmed
r/m, reg
reg,r/m
TEST acum, inmed Comparación.
r/m,inmed
r/m, reg
reg,r/m
WAIT Aguarda.
XCHG acum, reg Intercambio.
r/m,inmed
r/m, reg
reg,r/m
XLAT Traduce.
XOR acum, reg Operación XOR a nivel bit.
r/m,inmed
r/m, reg

reg,r/m

ARC-115
Para Laboratorio de Debug

Ejercicios 1 con DEBUG

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

SUMAS
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
```

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
```

ARC-115
Para Laboratorio de Debug

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
= 0707C000
(DXAX)

```
-rip
:100
-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
```

Para los cuatro ejercicios realizarlos con programa en debug de cada una de las operaciones.

Ejercicio 2 verificar que hace y modificarlo utilizando lenguaje ensamblador y ejecútelo en debug

a. 40

```
-rip
:100
-rax
:200
-rdx
:40
-e100
85.cd ED.21
b
-rip
:100
-rdx
:41
```

ARC-115

Para Laboratorio de Debug

Ejercicio 3 analice el siguiente programa y determine que hace.

```
-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
```

Ejercicio 4

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).

Ejercicio 5

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

Ejercicio 6

Haga un programa para que imprima el dígito más significativo de los siguientes números hexadecimales: 20_{16}, $A9_{16}$