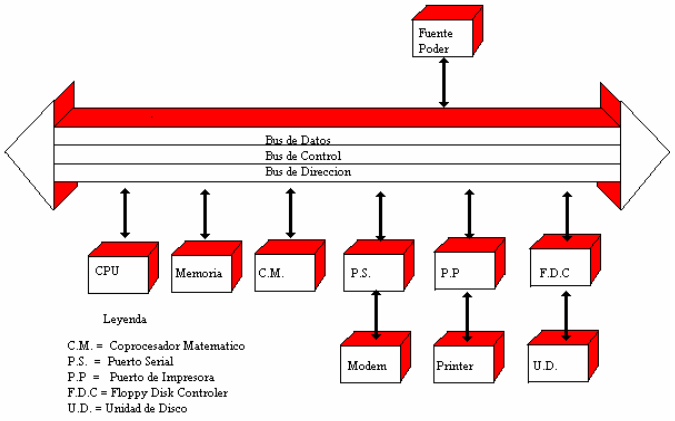
ESCUELA DE INGENIERÍA DE SISTEMAS
CÁTEDRA: ASSEMBLER

INVESTIGACIÓN

COMPONENTES BÁSICOS DE UN SISTEMA MS-DOS

Las operaciones de un sistema de computación incluyendo un IBM PC's y compatibles están basadas en un concepto simple. Ellas guardan intrusiones y datos en la memoria y usan el CPU para repetir intrusiones y datos recibidos desde la memoria y ejecutan las instrucciones para manipular los datos (Computadoras basadas en la Arquitectura de Von Newmann), por lo tanto el CPU y la memoria son los dos componentes básicos de cualquier sistema de computación. La memoria esta definida en dos variedades random access memory (**RAM)** la que permite la escritura y la lectura de cualquier localidad de memoria y la read only memory (**ROM)**, que la que contiene valores que pueden ser leídos pero no alterados. La ROM es usada para almacenar pequeños primitivos programas para ejecutar instrucciones de entrada y salida y control de periféricos. La RAM es usada para el Sistema Operativo y programas para usuarios. El Sistema Operativo es un componente fundamental en un sistema. Este programa de computadoras se toma la tarea de cargar otros programas y ejecutarlos, provee acceso a los archivos del sistema. La performa la E/S, y hace interfaces interactivas con el usuario. El sistema operativo es el que provee al sistema su personalidad. MS-DOS, OS/2, UNIX son ejemplo de algunos Sistema Operativos para PC, similarmente CP/M es un Sistema Operativos para antiguos



microprocesadores de INTEL de 8 Bits como el 8080. El hardware de toda

computadora incluyendo las computadoras que usan el MS-DOS está interconectados.

El CPU, memoria, y periféricos de entrada (teclado, escáner, lápiz óptico, lector de código de barra, micrófono, mouse etc.) y salida (monitor, impresora, cornetas, etc.) están todos interconectados por una serie de cables llamados Buces y cada Buz esta claramente definido. Un Buz es un hardware que especifica una señal y tiempo estándar que son seguidos y entendidos por el CPU y su circuito de soporte (incluyendo periféricos aun no instalados). Los buces a su vez se clasifican en Buz de Datos, Buz de Dirección, y Buz de Control. El Buz de Dirección selecciona la localidad de memoria entre la memoria usada y el CPU. El Buz de Control refiere la líneas de señales de tiempo y la línea de poder a los componentes. Un Sistema Operativo MS-DOS no necesita ningún tipo de Buz especifico. Esto es porque es posible tener el Buz en un segundo plano y estar ejecutando su sistema y así mismo lo aplica para otro tipo de programas compatibles. Por ejemplo los antiguos sistemas IBM PC-AT con los PC-AT Buces y el nuevo IBM PS/2 basado en la arquitectura de MCA (Micro Channel Architecture), y puede ser ejecutado en MS-DOS y OS/2.

ARQUITECTURA INTERNA DEL INTEL 80x86

Fue el primer microprocesador de 16 bits que INTEL fabrico a principios del año 1978. Los objetivos de la arquitectura de dicho procesador fueron los de ampliar la capacidad del INTEL 80x80 de forma simétrica, añadiendo una potencia de proceso no disponible en los micros de 8 bits. Algunas de estas características son: aritmética en 16 bits, multiplicación y división con o sin signo, manipulación de cadena de caracteres y operación sobre bits. También se han realizado mecanismo de software para la construcción de códigos reentrante y reubicable. Su estructura interna esta representada por la figura numero 1. Consta de 2 unidades claramente diferenciadas denominadas EU (Unidad de Ejecución) y BIU (interfaces del Bus).

La EU ejecuta las operaciones requeridas por la instrucciones sobre una UAL de 16 bits. No tiene conexión con el exterior y solamente se comunica con la BIU que es la parte que realiza todas las operaciones en el bus solicitadas por la EU. Un mecanismo, tal vez único dentro de los microprocesadores aunque muy empleado dentro de los mínimos y grandes ordenadores, es el denominado de búsqueda anticipada de instrucciones (prefetch). En el INTEL 8086 existe una estructura FIFO en RAM de 6 octetos de capacidad que es llenada por la BIU con los contenidos de las intrusiones siguientes a la que la EU esta ejecutando en ese momento.

Los registros del procesador se especifican en la figura numero 1. y son los siguientes:

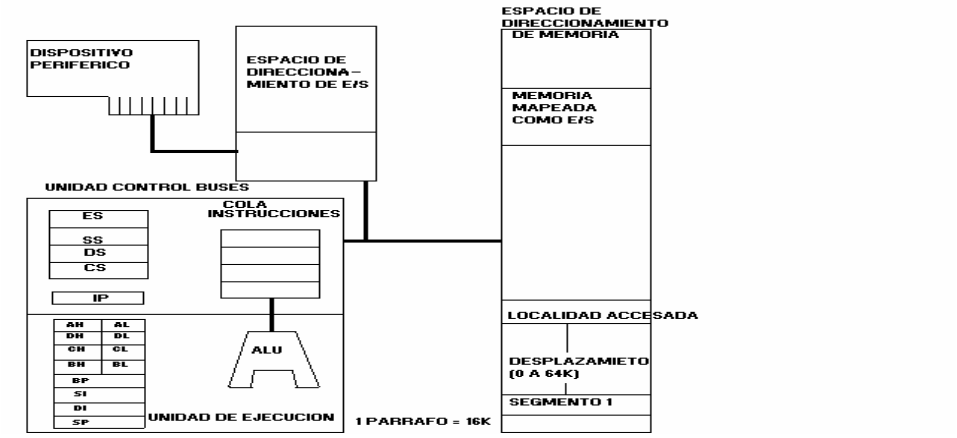
A.-) Cuatro registros de 16 bits, denominados AX,BX,CX y DX, que pueden ser direccionados de 8 registros de 8 bits, denominados AH,AL,....DL. los siete últimos son equivalentes a los registros A, H, L, B, C, D y E, del microprocesador Intel 80x86. El registro AX sirve fundamentalmente como acumulador y como registro de trasferencia en las intrusiones E/S. El registro BX puede usarse como acumulador y como registro base para calcular la direcciones de los datos de memoria. El registro CX puede usarse como acumulador y se utiliza como contador para las intrusiones interactivas. El registro DX puede usarse como acumulador y se emplea como puntero de datos en ciertas intrusiones especificas de E/S.

B.-) Cuatro registros de puntero de segmento denominado CS, DS, SS y ES. Dicho puntero definen cuatro segmentos de 64 K octetos cada uno. Cualquier dirección de memoria se forma, como ya veremos en el apartado de direccionamiento, sumando al puntero del segmento una dirección efectiva calculada por diversos procedimientos. El registro CS, (CODE SEGMENT) se usa junto con el PC para calcular las direcciones de las intrusiones del programa; el registro SS (STACK SEGMENT) se emplea junto con el SP (STACK POINTER) para calcular la dirección de las intrusiones que manejan la pila tales como PUSH, POP, CALL y RETURN; por su parte, el registro DS (DATA SEGMENT) se usa en instrucciones que manejan datos de memoria y el registro ES (EXTRA SEGMENT) se utiliza en instrucciones que manejan cadena de caracteres.

C.-) Cuatro registros que contiene direcciones de desplazamiento dentro de los segmentos denominadas SP, BP, SI, DI. El registro SP puntero de la pila los registros SI (INDEX SEGMENT) y DI (Índice Destino) contienen desplazamientos de los punteros de segmento DS y ES en las intrusiones que manejan cadena de caracteres. El registro BP (BASE POINTER) es el puntero base.

D.-) Un registro contador de programas, PC.

E.-) Un registro de estado, S, de 16 bits con la siguiente asignación: Bit b_o(C) es el acarreo, Bit b_2(P) es el de paridad, Bit b_4(A) es el de acarreo auxiliar, Bit b_6(Z) el de cero, Bit b_7(S) el de signo, Bit b_8 (T) el de Trap, Bit b_9(I) que sirve para controlar el bloqueo de las intrusiones, Bit b_{10}(D) que determinan si se han de autoincrementar o autodecrementar los punteros SI y DI en las intrusiones que manejan cadenas de caracteres, Bit b_{11}(O) que especifica el desbordamiento (Overflow).



El 8086 representa la arquitectura base para todos los microprocesadores de 16 bits de Intel: 8088, 8086, 80188, 80186 y 80286. Aunque han aparecido nuevas características a medida que estos microprocesadores han ido evolucionando; todos los procesadores Intel, usados en la actualidad en los PC's y compatibles son miembros de la familia 8086. El conjunto de instrucciones, registros y otras características son similares, a excepción de algunos detalles, todo la familia 80x86 en adelante poseen dos características en común: ***A)*** **Arquitectura Segmentada,** Esto significa que la memoria es divida en segmentos con un tamaño máximo de 64k (información importante para el direccionamiento de la memoria en la futura programación segmentada en el lenguaje ensamblador) y ***B)*** **Compatibilidad** de Las intrusiones y registros de las anteriores versiones son soportados por las nuevas versiones, y estas versiones son soportadas por versiones anteriores.

La familia de microprocesadores 80x86 consta de los siguientes microprocesadores:

8088: Es un microprocesador de 16 bits, usado en las primeras PC´S (XT compatibles). Soporte solamente el modo real. Es capaz de direccionar un megabytes de memoria y posee un bus de datos de 8 bits. El ***8086*** es Similar al 8088, con la excepción de que el bus de datos es de 16 bits. El ***0188*** es similar al 8088, pero con un conjunto de instrucciones extendidos y ciertas mejoras en la velocidad de ejecución. Se incorporan dentro del microprocesador algunos chips que anteriormente eran externos, consiguiéndose unas mejoras en el rendimiento del mismo. El ***80186*** es igual al 80188 pero con un bus de datos de 16 bits. El ***80286*** Incluye un conjunto de instrucciones extendidos del 80186, pero además soporta memoria virtual, modo protegido y multitarea. El ***80386*** soporta procesamientos de 16 y 32 bits. El 80386 es capas de manejar memoria real y protegida, memoria virtual y multitarea. Es más rápido que el 80286 y contiene un conjunto de

instrucciones ampliables. El ***80386SX*** es similar al 80386 por un bus de datos de solo 16 bits. El ***80486*** incorpora un cache interno de 8k y ciertas mejoras de velocidad con respecto al 80386. Incluye un coprocesador matemático dentro del mismo chip. El ***80486SX*** es Similar a los 80486 con la diferencia que no posee coprocesador matemático y ***80486DX2*** es Similar al 80486, pero con la diferencia de que internamente, trabaja al doble de la frecuencia externa del reloj.

El 80x86 tiene dos procesadores en el mismo chip. Estos son **La Unidad de Ejecución** y **La Unidad de Interface con los Buces**. Cada uno de ellos contiene su propio registro, su propia sección aritmética, sus propias unidades de control y trabajan de manera asincrónica el uno con el otro para proveer la potencia total de computo. La unidad de Interface de bus se encarga de buscar las instrucciones para adelantar su ejecución y proporciona facilidades en el manejo de las direcciones. Luego, la unidad de Interface se responsabiliza del control de la adaptación con los elementos externos del CPU central. Dicha unidad de interface proporciona una dirección de 20 Bits o un dato de 16 para la unidad de memoria o para la unidad de E/S en la estructura externa del computador.

DEFINICIÓN DE LOS SISTEMAS NUMÉRICOS

SISTEMA DECIMAL

Desde hace muchos años, el hombre ha utilizado como sistema para contar el denominado decimal, que derivó del sistema numérico indoarábigo; posiblemente se adoptó este mismo por contar con diez dedos en las manos. El sistema decimal es unos de los denominados sistemas posicionales, utilizando un conjunto de símbolos cuyo significado depende fundamentalmente de su posición relativa al símbolo de coma (.), Denominado coma decimal, que en caso de ausencia se supone colocada implícitamente a al derecha. Utiliza como base el 10, que corresponde al número de símbolos que comprende para la representación de cantidades; estos símbolos (también denominados dígitos) son: ***0 1 2 3 4 5 6 7 8 9.*** Una determinada cantidad, que denominaremos número decimal, se puede expresar de la siguiente forma:

$$N^0 = \sum_{i=-d}^{n} (dígito)_i \times (base)^i$$

Donde: base = 10, i= posición respecto a la coma, D = Nº. Dígitos a la izquierda de la coma, N = Nº , de dígitos a la Izquierda de la coma, Dígito = cada uno de los que componen el número. Esta forma corresponde al teorema fundamental de la numeración y por tanto corresponde a la representación,Por ejemplo, la representación de las cantidad 1992 es:

$$1992 = 1 \times 10^3 + 9 \times 10^2 + 9 \times 10^1 + 2 \times 10^0$$

SISTEMA BINARIO

Es el sistema que utiliza internamente el hardware de las computadoras actuales, por ello será el sistema al que se le prestará mayor atención y estudio. Se basa en la representación de cantidades utilizando los dígitos 1 y 0. Por tanto su base es 2 (número de dígitos del sistema). Cada dígito de un número en este sistema se denomina bit (contracción de **bi**nary digi**t**).

Se puede utilizar con nombre propio determinados conjuntos de dígitos en binario. Cuatro bits se denominan **cuaterno** (ejemplo: 1001), ocho bits **octeto** o **byte** (ejemplo: 10010110), al conjunto de 1024 bytes se le llama **Kilobyte** o simplemente **K,** 1024 Kilobytes forman un **megabyte** y 1024 megabytes se denominan **Gigabytes.**

SISTEMA OCTAL

Es un sistema de numeración cuya base es 8, es decir, utiliza 8 símbolos para la representación de las cantidades, Estos símbolos son; ***0 1 2 3 4 5 7***. Este sistema también es de los llamados posicionales y la posición de sus cifras se mide con relación a la coma decimal que en caso de no aparecer se supone implícitamente a la derecha del número. La aritmética en este sistema es similar a la de los sistemas binario y decimal, por lo que no entraremos en su estudio.

SISTEMA HEXADECIMAL

Es un sistema posicional de numeración en el que su base es 16, por tanto, utilizará 16 símbolos para la representación de cantidades, estos símbolos son; ***0 1 2 3 4 5 6 7 8 9 A B C D E F***. Se la asignan los siguientes valores absolutos a los símbolos A, B, C, D, E y F:

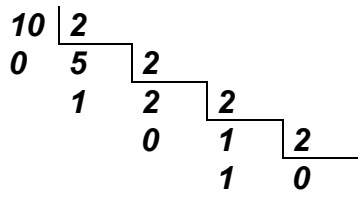
Símbolo	***Valor Absoluto***
A	10
B	11
C	12
D	13

E F	14 15

Cabe destacar que este sistema numérico tiene mucha utilidad, en las operaciones internas del computador, ya que por ejemplo cuando se utiliza el comando DEBUG, los valores contenidos en todos los registros de memoria se especifican en hexadecimal. Sí dos valores se suman, restan, multiplican o dividen, el resultado se presenta en hexadecimal. Dado que lo más común es el sistema decimal, es necesario efectuar una conversión de hexadecimal y viceversa para obtener el resultado de cualquier operación en el formato DEBUG.

TRANSFORMACIONES

- ***Conversión Decimal-Binario:*** Para convertir números enteros de decimal a binario, la forma más simple es dividir sucesivamente el numero decimal y los cocientes que se van obteniendo por, hasta que una de las divisiones se haga 0. La unión de todos los restos obtenidos escritos en orden inverso, nos proporcionan el número inicial expresado en el sistema binario. **Ej.:**



$10_{(10)}=1010_{(2)}$

- ***Conversión de una fracción decimal a binario:*** la forma más simple consiste en multiplicar dicha fracción por 2,obteniendo en la parte entera del resultado el primero de los dígitos binarios de la fracción binaria que buscamos. A continuación repetimos el mismo proceso con la parte fraccionaria del resultado anterior, obteniendo en la parte entera del nuevo resultado el segundo de los dígitos buscados. Iteramos sucesivamente de esta forma, hasta que desaparezca la parte fraccionaria o hasta que tengamos los suficientes dígitos binarios que nos permitan no sobrepasar un determinado error.

- ***Conversión de binario a decimal:*** el método consiste en reescribir él numero binario en posición vertical de tal forma que la parte de la derecha quede en la zona superior y la parte izquierda quede en la zona inferior. Se repetirá el siguiente proceso para cada uno de los dígitos

comenzados por el inferior: Se coloca en orden descendente la potencia de 2 desde el cero hasta n, donde el mismo el tamaño del numero binario, el siguiente ejemplo ilustra de la siguiente manera. Utilizando el teorema fundamental de la numeración tenemos que 1001.1es igual a:

$$1*2^3+0*2^2+0*2^1+1*2^0+1*2^{-1}=9.5_{(10)}$$

- ***Conversión decimal – octal:*** Consiste en dividir un numero y sus sucesivos cocientes obtenidos por ocho hasta llegar a una división cuyo cociente sea 0. El numero Octal buscado es el compuesto por todos los restos obtenidos escritos en orden inverso a su obtención. ***Ej.:***

1992	***8***		
39	***249***	***8***	
72	***09***	***31***	***8***
0	***1***	***7***	***3***

$1000_{(10)}=3710_{(8)}$

- ***Conversión de una fracción decimal a una octal:*** Se toma la fracción decimal y se multiplica por 8, obteniendo en la parte entera del resultado el primer dígito de la fracción octal resultante y se repite el proceso con la parte decimal del resultado para obtener el segundo dígito y sucesivos. El proceso termina cuando desaparece la parte fraccionaria del resultado o dicha parte fraccionaria es inferior al error máximo que deseamos obtener. ***Ej. :***

0.140625*8=1.125

$0.140625_{(10)}=0.11_{(8)}$

0.125*8=1.0

- ***Conversión octal a decimal:*** Existen varios métodos siendo el más generalizado el indicado por el TFN (Teorema fundamental de la numeración) que hace la conversión de forma directa por medio de la formula. ***Ej. :*** utilizando el teorema fundamental de la numeración tenemos que 4701 es igual a:

$$4*8^3+7*8^2+0*8^1+1*8^0=2497_{(10)}$$

- ***Conversión decimal – hexadecimal:*** Se divide el numero decimal y los cocientes sucesivos por 16 hasta obtener un cociente igual a 0. El número hexadecimal buscado será compuesto por todos los restos obtenidos en orden inverso a su obtención. ***Ej.:***

1000	***16***	
40	***62***	***16***
8	***14***	***3***

$1000_{(10)}=3E8_{(16)}$

- ***Conversión de una fracción decimal a hexadecimal:*** a la fracción decimal se multiplica por 16, obteniendo en la parte entera del resultado el primer dígito de la fracción hexadecimal buscada, y se repite el proceso con la parte fraccionaria de este resultado. El proceso se acaba cuando la parte fraccionaria desaparece o hemos obtenido un número de dígitos que nos permita no sobrepasar el máximo error que deseemos obtener. ***Ej.:*** Pasar a hexadecimal la fracción decimal 0.06640625

  0.06640625*16=1.0625

  0.0625*16 = 1.0 Luego

  $0.06640625_{(10)}=0.11_{(16)}$

- ***Conversión hexadecimal - decimal:*** el método más utilizado es el TFN que nos da el resultado por la aplicación directa de la formula. ***Ej. :*** utilizando el teorema fundamental $2*16^2+C*16^1+A*16^0=714_{(10)}$ de la numeración tenemos que 2CA es igual a:

- ***Conversión de hexadecimal-binario***: para convertir un numero hexadecimal a binario, se sustituye cada dígito hexadecimal por su representación binaria según la siguiente tabla.

Dígito Hexadecimal	***Dígito Binarios***
0	***0000***
1	***0001***
2	***0010***
3	***0011***
4	***0100***
5	***0101***
6	***0110***
7	***0111***
8	***1000***
9	***1001***
A	***1010***
B	***1011***
C	***1100***
D	***1101***
E	***1110***
F	***1111***

Ej.: pasar el número 2BC a binario

2	***B***	***C***

0010	***1011***	***1100***

Finalmente él número hexadecimal en binario es igual a: 001010111100

- ***Conversión de octal a binario:*** para convertir un numero octal a binario se sustituye cada dígito octal en por sus correspondientes tres dígitos binarios según la siguiente tabla.

Dígito Octal	***Dígito Binario***
0	***000***
1	***001***
2	***010***
3	***011***
4	***100***
5	***101***
6	***110***
7	***111***

Ej.: Convertir el número octal 1274 en binario.

1	***2***	***7***	***4***
001	***010***	***111***	***100***

Por lo tanto el número octal en binario es igual a: 001010111100

TIPOS DE PROGRAMAS EJECUTABLES

- **ESTRUCTURA DEL PROGRAMA CON EXTENSIÓN COM:** Un programa con extensión COM están almacenados en archivos que contienen una copia fiel del código a ser ejecutado. Ya que no contienen información para la reasignación de localidades, son más compactos y son cargados más rápidamente que sus equivalentes EXE. El MS-DOS no tiene manera de saber si un archivo con extensión COM es un programa ejecutable válido. Este simplemente lo carga en memoria y le transfiere el control. Debido al hecho de que los programas COM son siempre cargados inmediatamente después del PSP y no contienen encabezado que especifique el punto de entrada al mismo, siempre debe comenzar en la dirección 0100h. Esta dirección deberá contener la primera instrucción ejecutable. La longitud máxima de un programa COM es de 65536 bytes, menos la longitud de PSP (256 bytes) y la longitud de la pila (mínimo 2 bytes).

Cuando el sistema operativo transfiere el control a un programa COM, todos los registros de segmento apuntan al PSP. El registro apuntador de pila (SP), contiene el valor en la memoria de OFFFEh si la memoria los permite. En otro caso adopta el mínimo valor posible menos dos bytes (el MS-DOS

introduce un cero en la pila antes de transferir el control al programa). Aún cuando la longitud de un programa COM no puede exceder de los 64 , las versiones actuales del MS-DOS reservan toda la memoria disponible. Si un programa COM debe ejecutar otro proceso, es necesario que el mismo libere la memoria no usada de tal manera que pueda ser empleada por otra aplicación. Cuando un programa COM termina, puede retornar al control del sistema operativo por varios medios. El método preferido es el uso de la función **4Ch** de la **Int 21**, la cual permite que el programa devuelva un código de retorno al proceso que invocó. Sin embargo, si el programa está ejecutándose bajo la versión 1.00 del MS.DOS, el control debe ser retornado mediante el uso de la **Int 20h**. Un programa COM puede ser ensamblado a partir de varios módulos objeto, con la condición de todos ellos empleen los mismos nombres y clases de segmentos y asegurando que él modulo inicial, con el punto de entrada en 0100h sea enlazado primero. Adicionalmente todos los procedimientos y funciones deben tener el atributo NEAR, ya que todo el código ejecutable estará dentro del mismo segmento.

Al enlazar un programa COM el enlazador mostrará el siguiente mensaje; "***Warnig: no stack segment***". Este mensaje puede ser ignorado, ya que el mismo se debe a que se ha instruido al enlazador para que genere un programa con extensión EXE donde el segmento de pila debe ser indicado de manera explícita, y no así en los COM donde esta es asumida por defecto. En la zona desde 000Ah hasta 0015h dentro del PSP se encuentran las direcciones de las rutinas manejadoras de los eventos Ctrl-C y Error critico. Si el programa de aplicación altera estos valores para sus propios propósitos, el MS-DOS los restaura al finalizar la ejecución del mismo.

Estructura del prefijo de programa.

0000h	***INT 20***
0002h	***Segmento, final del bloque de asignación***
0004h	***Reservado***
0005h	***Invocación FAR a la función despachadora del MS-DOS***
000Ah	***Vector de interrupción de terminación (Int22h)***
000Eh	***Vector de interrupción Ctrl-C (Int23h)***
0012h	***Vector de interrupción de error crítico (Int24h)***
0016h	***Reservado***
002Ch	***Segmento de bloque de variables de ambiente***
002Eh	
005Ch	***Bloque de control de archivo por defecto (#1)***
006Ch	***Bloque de control de archivo por defecto (#2)***
0080h	***Líneas de comandos y área de transferencia de disco***
00FFh	***Final del PSP***

La palabra de datos en desplazamiento 002Ch contiene la direccion del segmento de bloque de variables de ambiente (Environment block), el cual contiene una serie de cadenas ASCIIZ. Este bloque es heredado del proceso que causo la ejecución del programa aplicación. Entre la información que contiene tenemos, el paso usado por el COMAND.COM para encontrar el archivo ejecutable, el lugar del disco donde se encuentra el propio COMAND.COM y el formato del prompt empleado por este. La cola de comandos, la cual está constituida por los caracteres restantes en la línea de comandos, después del nombre del programa, es copiado a partir de la localidad 0081h en el PSP. La longitud de la cola, sin incluir el carácter de retorno al final, está ubicada en la posición 0080h. Los parámetros relacionados con redireccionamiento o piping no aparecen en esta posición de la linea de comandos, ya que estos procesos son transparentes a los programas de aplicación. Para proporcionar compatibilidad con CP/M, el MS-DOS coloca los dos primeros comandos en la cola, dentro de los bloques de control del archivo (FCD) por defecto en las direcciones PSP:005Ch yPSP:006Ch asumiendo que pueden ser nombre de archivos. Sin embargo, si alguno de estos comandos son nombres de archivos que incluyen especificaciones del paso, la información colocada en los FCB no será de utilidad ya que estas estructuras no soportan el manejo de estructuras jerárquicas de archivos y subdirectorios. Los FCB son de muy escaso uso en los programas de aplicación modernos. El área de 128 bytes ubicado entre las direcciones 0080h y 00FFh en el PSP pueden también servir como área de transferencia de disco por defecto (DTA), la cual es establecida por el MS-DOS antes de transferir el control al programa de aplicación. A menos que el programa establezca de manera explícita otra DTA, este será usado como buffer de datos para cualquier intercambio con disco que este efectué. Los programas de aplicación no deben alterar la información contenida en el PSP a partir de la dirección 005Ch.

- **ESTRUCTURA DE UN PROGRAMA DE EXTENSION EXE:** Los programas EXE son ilimitados en tamaño (él limite lo dictamina la memoria disponible del equipo). Además, los programas EXE pueden colocar el código, datos y pila en distintos segmentos de la memoria. La oportunidad de colocar las diversas partes de un programa en fragmentos diferentes de memoria y la de establecer segmentos de memoria con solamente códigos de que pudieran ser compartidos por varias tareas, es un significativo para ambientes multitareas tales como el Microsoft Windows. El cargador del MS-DOS, sitúa al programa EXE, inmediatamente después del PSP, aunque el orden de los segmentos que lo constituyen pueden variar. El archivo EXE contiene un encabezado, bloque de información de control, con un formato característico. El tamaño de dicho encabezado pude variar dependiendo del numero de instrucciones que deben ser localizadas al momento de carga del programa, pero siempre será múltiplo de 512. Antes de que el MS-DOS

transfiera el control al programa, se calculan los valores iniciales del registro del segmento de código (CS) y el apuntador de instrucciones (IP) basados en la información sobre el punto de entrada, al programa, contenida en el encabezado del archivo EXE. Esta información es general a partir de la instrucción END en él modulo principal del programa fuente. Los registros de segmentos de datos y segmentos extras inicializados de manera que apunten al PSP de tal manera que el programa pueda tener acceso a la información contenida.

Imagen de Memoria de un programa EXE típico

SS:SP	***Segmento de Pila***
SS:0000h	***Datos del Programa***
CS:0000h	***Código del Programa***
DS:0000h	***Prefijo del segmento del Programa***
ES: 0000h	

Formato de un archivo de carga EXE.

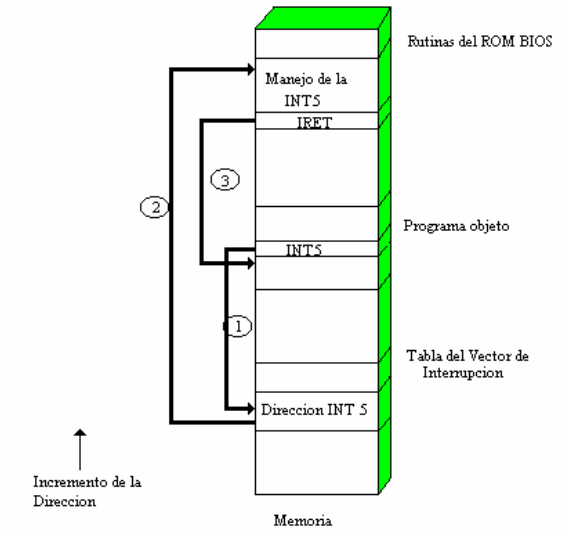
0000h	***Primera parte del identificador del archivo EXE (4Dh)***
0001h	***Segunda parte del identificador de archivo EXE (5Ah)***
0002h	***Longitud del archivo MOD 512***
0004h	***Tamaño del archivo, en páginas de 512 bytes, incluyendo encabezado***
0008h	***Número de ítems en la tabla de relocalizaciones***
000Ah	***Tamaño del encabezado en párrafos (16 bytes)***
000Ch	***Número mínimo de párrafos requeridos para el programa***
000Eh	***Máximo número de párrafos deseables para el programa***
0010h	***Desplazamiento del segmento del módulo de pila***
0012h	***Suma de chequeo***
0016h	***Contenido del apuntador de instrucciones al comenzar al el programa***
0018h	***Desplazamiento del segmento del módulo de código***
001Ah	***Desplazamiento del primer ítem en la tabla de relocalizaciones***
001Bh	***Número de overplay (0 para la parte residente del programa)***
	Tabla de relocalizaciones
	Espacio reservado (longitud variable)
	Segmento de programas y datos
	Segmento de pila

El contenido inicial del segmento de pila y de la apuntador de pila provienen también del encabezado del archivo. Esta información es derivada de la declaración del segmento de pila efectuada mediante la sentencia STACK. El espacio reservado para la pila puede ser inicializado o no dependiendo de la manera como este haya sido declarado. Puede ser conveniente en muchos casos inicializar el segmento de pila con un patrón de caracteres predeterminados que permitan su posterior inspección. Cuando el programa EXE finaliza su ejecución debe retornar el control al sistema operativo mediante la función 4Ch de la Int 21h. Existen otros métodos, pero no ofrecen ninguna otra ventaja y son considerablemente menos convenientes " Generalmente requieren que el registro CS apunte al segmento de PSP".

Un programa EXE puede ser construido a partir de varios módulos independientes. Cada modulo puede tener nombres diferentes para el segmento de código y los procedimientos pueden llevar el atributo NEAR o FAR, dependiendo del tamaño del programa ejecutable. El programador debe asegurarse de que los módulos, a ser enlazados solo tenga una declaración de segmento de pila y que haya siod definido un único punto de entrada (por medio de la directiva END). La salida del enlazador es un archivo con extensión EXE el cual puede ser ejecutado inmediatamente.

ACCESO A LAS INTERRUPCIONES DEL BIOS Y DOS DESDE ROM

El ROM BIOS y DOS contiene rutinas que pueden ser usadas en los programas. Estas rutinas usualmente no son invocadas por procedimientos usuales, pero pueden ser accedido por mecanismos de interrupción. La mayoría de los programadores típicamente organizan los programas por intrusiones CALL. El BIOS y las funciones del DOS están en forma de código objeto, y se encuentran en direcciones de memoria, en el lenguaje ensamblador hay una instrucción denominada *INT* que genera una interrupción de software, en un microprocesador 80x 86 que provee una solución a determinado código de interrupción. El 80 x 86 usa código de interrupciones como índice en una tabla para localizar la rutina a ejecutar



cuando la interrupción ocurre. Esta tabla de funciones son conocidas como Tabla del Vector de Interrupción (IVT) y las funciones son conocidas como Interrupciones Rutinarias de Servicio (ISR's). El IVT esta localizado en el primer 1,024 Byte de Memoria y contiene 256 entradas. Desde cada dirección ISR es de la forma CS:IP cada entrada en el IVT requiere de 4 Byte de almacenamiento (256 * 4 = 1,024 B). El 80x86 recibe la señal de interrupción primero empuja (PUSH) los Flags, CS y el registro IP que se encuentra en la pila en ese orden, luego el CPU usa el numero de interrupción para indexarlo en el vector de interrupción (IVT) y luego salta a las rutinas de servicio de interrupción (ISR's) para esa interrupción. El ISR's termina con IRET (Interrup RETRY) los cual remueve los datos de la pila (POP) el **I**ntrusión **P**ointer (IP), el **C**ode **S**egment (CS) y Flags de la Stack (pila) por la cual retorna el control a la interrupción del programa. Ej: 1.- Ejecutando la interrupción 5 ocasiona que el microprocesador grabe el siguiente estado y salta a la función de la tabla IVS en la entrada de la interrupción 5, 2.- El microprocesador ejecuta el código que maneja en esa interrupción (imprimir pantalla), 3.- Cuanto IRET es ejecutado se devuelve el control justo después del comando colocado en el programa objeto.

Principales Interrupciones del BIOS y del DOS

INT	***TIPO***	***DESCRIPCIÓN***
2	BIOS	Este tipo de interrupción no se puede evitar. Utiliza el BIOS NEM2, procedimiento NMI-INT y aparece cuando se detectan errores en la memoria sobre la tarjeta del sistema (Parity Check 1) o se tiene problemas con tarjetas que se añaden al sistema (Parity Check2)
5	BIOS	Esta interrupción se encarga de imprimir el contenido de la pantalla bajo el control del programa. EL llamado al procedimiento tipo FAR en PRINT SCREEN y la dirección 0050;0000 contiene el estado
8	BIOS	Esta rutina maneja la interrupción del temporizador proveniente del canal 0 del temporizador 8253. La rutina lleva el conteo del numero de interrupciones desde que se energizó la computadora.
9	BIOS	Esta rutina es un procedimiento FAR KB-INT. La rutina continua en la dirección F000;EC32 y constituye la interrupción del teclado. La INT 16h es la rutina de E/S del teclado y es más flexible.
E	BIOS	Este procedimiento de tipo FAR, DISK-INT maneja la interrupción del diskette.
F	DOS	Activa la misma llamada que type 4.
10	BIOS	El conjunto de rutinas asociados con este procedimiento NEAR VIDEO-E/S, constituye la interfaz con el TRC.
11	BIOS	El procedimiento proporciona él numero de puertos para

		la impresora, adaptadores de juegos, interfaces RS-232C, numero de unidades de, Diskettes, modos de video y tamaños del RAM
12	BIOS	Proporciona el tamaño de la memoria
13	BIOS	Llama a varias rutinas para llevar operaciones de entrada y salidas del disco.
14	BIOS	Este procedimiento permite al usuario la entrada y salida de datos desde el puerto de comunicaciones desde el puerto de comunicaciones RS-232C.
15	BIOS	Interrupción empleada para controlar las operaciones de E/S en casettes.
16	BIOS	Esta interrupción utiliza a AX para leer el teclado.
17	BIOS	Esta rutina proporciona la comunicación con la impresora. Los parámetros necesarios son colocados en los registros AX y DX.
18	BIOS	Esta interrupción llama al casette de basic.
19	BIOS	La rutina asociada con esta interrupción, lee el sector uno de la pista cero del disco en la unidad A, a la que le transfiere el control
1A	BIOS	Esta rutina permite seleccionar o leer el contenido del reloj que lleva la hora. El registro CX contiene la palabra más significativa del conteo mientras que en DX se encuentra la menos significativa.
1B	DOS	Esta interrupción se presenta cada vez que se genera una interrupción proveniente del teclado.
1C	BIOS	Esta interrupción provoca la ejecución IRET.
1D	BIOS	Esta tabla de bytes y rutinas necesarias para establecer varios parámetros para gráficos.
1E	DOS	Tabla de Diskette.
1F	DOS	Tabla de gráficos.
20	DOS	Esta interrupción es generada por DOS para salirse un programa, es la primera dirección del área correspondiente al segmento prefijo del programa.
21	DOS	Esta interrupción consta de varias opciones, una de ellas es solicitar funciones.
22	DOS	Cuando termina la ejecución de un programa esta interrupción transfiere el control a la dirección especificada por el vector de interrupción. Esta interrupción nunca debe generarse de manera directa.
23	DOS	Esta interrupción es generada como respuesta a un CRTL BREAK.
24	DOS	Esta interrupción se llama cada ves que ocurre un error crítico dentro de dos, como puede ser un error de disco.

25	DOS	Esta interrupción transfiere el control, para lectura, al manejador del dispositivo (driver).
26	DOS	Esta interrupción transfiere el control, para escritura, a manejador del dispositivo.
27	DOS	Este vector es empleado, para que al término de un programa este permanezca residente en la memoria del sistema una vez que DOS toma de nuevo el control.
2F	DOS	Esta interrupción define una interfaz general entre dos procesos, él numero especificado en Ah indica a cada manejador y Al contiene la función del manejador.

ANEXO Nº 1

BIBLIOGRAFÍA

- Medina, Ramon. ***PROGRAMACIÓN AVANZADA EN LENGUAJE ENSAMBLADOR***. 1992, Págs. 7-19.
- Barbakati, Nabaiyoti. Haybe, Randal. ***THE WAITE GROUP´S MICROSOFT MACRO ASSEMBLER BIBLE***.
- Godfrey, Terry J.***LENGUAJE ENSAMBLADOR PARA MICROCOMPUTADORAS IBM(PARA PRINCIPIANTES Y AVANZADOS)***. 1991, Editorial PRENTICE-HALL HISPANOAMERICANA, S.A. Págs. 8-12 y 143-148.
- Alcalde E.; García M.; Peñuelas S. ***INFORMÁTICA BÁSICA***.1988, Editorial McGraw-Hill. Págs. 23-48.

Trabajo realizado por:
Miguel Pita
M_PITA@HOTMAIL.COM