TEMA 1. CARACTERÍSTICAS DEL 8086.

ORGANIZACIÓN DE LA MEMORIA.

ALU de 16 bits.

En el 8086 tenemos DBUS de 16 bits.

Celdillas de memoria de 8 bits.

No se emplean celdillas de memoria de 16 bits debido a que:

- Una CPU no puede acceder directamente a media celdilla, sino que debe acceder como mínimo a una celdilla completa.
- Es conveniente para muchas cosas trabajar byte a byte, por ejemplo con caracteres ASCII (8 bits).
- Si las celdillas fuesen de distinto tamaño en distintos micros debido a que estos tienen DBUSes de diferente tamaño, sería problemático, pues cada uno direccionaría de una forma diferente.

Fig. 1 Esquema de buses del 8086.

Memoria Direcciones Pares

A0 D0−D7

A1−A19

D8−D15

Decodificador

Direcciones Memoria Direcciones Impares

BHE

D0−D15

A0−A19 = Líneas del ABUS (Bus de Direcciones)

D0−D15 = Líneas del DBUS (Bus de Datos)

BHE = Byte High Enable

Ej: Llega al bus de direcciones una dirección PAR: 8 4 A 7 C h 1000 0100 1010 0111 1100

Siguiente dirección IMPAR: 8 4 A 7 D h 1000 0100 1010 0111 1101

Se diferencian en A0

La señal A0 del ABUS va por otro cable (no por el que van las señales A1 − A15) debido a que un número par y su siguiente sólo se diferencian en ese A0.

Posibilidades:

Dirección A0 BHE Situación.

PAR (A0=0)	0	0	Se accede a WORD (16 bits):	Dirección PAR por D0−D7			Siguiente dirección (IMPAR) por D8−D15
		0	1	Se accede a BYTE (8 bits):	Dirección PAR por D0−D7 {La siguiente dirección (IMPAR) no produce señal pues la puerta triestado lo impide.}		
	IMPAR (A0=1)	1	0	Acceso a BYTE (8 bits):	Dirección IMPAR por D8−D15		
		1	1	NO SE USA (se inhabilitan las 2 puertas triestado).			

TIPOS DE DATOS.

Enteros Sin Signo	Binario Natural:	Tamaños BYTE, WORD y DOUBLE WORD		
		BCD 8421	Empaquetado (Tamaño Byte)	
				Desempaquetado (Tamaño Byte)
Enteros Con Signo	Convenio de CA2	Tamaños BYTE, WORD y DOUBLE WORD.		

BYTE 8 bits

WORD 16 bits

DWORD 32 bits

BCD Empaquetado: 1000 1010

8 5 (85 decimal)

BCD Desempaquetado: x x x x 0 1 1 0

Sin significado Cifra BCD (6 en decimal)

El BCD desempaquetado, que a priori parece poco efectivo, sirve p.e. cuando tenemos un código alfanumérico (ASCII, EBCDIC...), puede surgir la cuestión: el código 0 ASCII, tiene el valor 30h, el 1 ASCII tiene el 31h, etc...

ASCII

0 0011 0000

- 0011 0001

...

... 0011 1001

Es una manera de que el micro haga operaciones directamente con valores ASCII.

PARTICULARIDADES:

- **Solapamientos:** Diferentes direcciones lógicas pueden apuntar a la misma dirección física.

Ej:

348Ah : 5FDEh = 3A87Eh

3A87h : 000Eh = 3A87Eh

3A80h : 007Eh = 3A87Eh

Dirección Lógica Dirección Física

- El Segmento es un valor que no puede especificarse de manera directa en una instrucción, sino que debe hacerse mediante los REGISTROS DE SEGMENTO (el segmento debe estar previamente colocado en un registro de segmento).
- Cuando un offset se desborda (pasa de FFFFh al siguiente), pasa a la primera posición del mismo segmento (0000h), es decir, es cíclico. El acarreo del offset no modifica el segmento.
- Si tengo un segmento que empieza casi al final de la memoria (es decir, queda menos de 64 KB), el segmento continuaría al principio de la memoria.

ALINEAMIENTO DE DATOS E INSTRUCCIONES.

Para leer una WORD que empiece en dirección par, comienza en esa dirección par y lee 2 bytes (un word) de una vez.

Para leer una WORD que empiece en dirección impar, lee los 2 bytes en 2 veces, primero un byte y luego el siguiente.

Si el WORD comienza en dirección par, se dice que está ALINEADA a word:

84A7Ch Par WORD Alineada

84A7Dh Impar

Si el WORD comienza en posición impar, no está alineada:

84A7Dh Impar WORD No Alineada

84A7Eh Par

DIRECCIONES DE MEMORIA.

Dirección Física: La que usa el Hardware (número de 20 bits que circula por el ABUS, que va desde

el 00000h hasta el FFFFFh).

Dirección Lógica: La que se maneja desde el programa. En el programa, para referenciar una posición

de memoria física, lo haremos mediante la técnica de Segmentación, usando para ello dos números de 16 bits: segmento y offset.

Segmentación:

Segment : Offset

Tanto el Segmento como el Offset son números de 16 bits que nos sirven para realizar una transformación de direcciones:

DIR_FISICA = (16*Segmento + Offset)

10h

Ej:

Dir. Lógica 348Ah : 5FDEh

Dir. Lógica Dir. Física

10h*348Ah = 348A0h 348Ah : 5FDEh 3A87Eh

5FDEh = 5FDEh

3A87Eh Dir. Física

Segmento:

Una dirección de segmento indica un área de memoria de 64 KB (Segmento de memoria), que comienza en la dirección física 10h*Segmento (es decir, 16d*Segmento).

Advertencia: Segmento de memoria < > Segmento de ensamblador.

Offset:

El Offset es la dirección relativa (desplazamiento) con relación al principio del Segmento.

Ej:

Memoria

0000h : 0000h

Offset

348Ah : 0000h = 348A0h 0000h

Segmento ...

de 64 KB

Memoria ...

348Ah : FFFFh = 348Ah + FFFFh FFFFh

FFFFh : FFFFh

REGISTROS DEL PROCESADOR 8086.

GENERALES:

Son cuatro registros que se denominan AX, BX, CX y DX. Cada uno de estos registros se subdivide además en dos registros de tipo byte, que se denominan con la misma letra que el registro completo pero terminada en H si se refiere al byte superior o HIGH (8 bits más significativos del registro) o en L si se refiere al byte inferior o LOW (los 8 bits menos significativos). Así pues, si hay un dato de tipo word almacenado en el registro AX, es posible acceder a su byte superior mediante el registro AH y al inferior a través de AL. Los otros tres registros también cuentan con esta propiedad, por lo que existe un BL y BH, CL y CH y DL y DH.

El registro AX (Acumulador) es utilizado por algunas operaciones aritméticas como origen o destino de los datos de forma implícita, es decir, que siempre utilizan este registro y no hay posibilidad de cambiarlo por otro.

El registro BX (Base) se utiliza en algunos modos de direccionamiento para formar la dirección de memoria de la que obtiene o en la que almacena los datos una determinada instrucción.

El registro CX (Contador) se utiliza con las instrucciones de repetición y de bucle, almacenando el número de veces que se repetirá una determinada instrucción o fragmento de programa.

El registro DX (Datos) se combina con AX en algunas instrucciones que manejan cantidades de 32 bits (DX&AX) y especifica la dirección de un puerto de entrada/salida.

INDICES:

Su principal finalidad es la de almacenar la posición de memoria donde se encuentra algún dato necesario para las instrucciones del programa. Los registros índices son SI y DI.

Los registros SI (Source Index) y DI (Destination Index) se utilizan como índices en los fragmentos de programa que necesitan acceder a varias posiciones consecutivas de memoria. SI es utilizado por algunas instrucciones como el origen de los datos que necesita, mientras que DI de utiliza como destino de los resultados que se producen.

DE PILA:

El registro SP (Stack Pointer, Puntero de Pila) contiene el puntero de la pila, apuntando (almacenando la dirección de memoria) donde se guardará el siguiente dato en la pila.

El registro BP (Base Pointer) es un registro especialmente diseñado para realizar lecturas/escrituras en la Pila.

Hay una forma de pasar parámetros a un procedimiento a través de la Pila:

- Colocamos en la Pila los datos a pasar.
- Apuntamos con BP al primer elemento.
- Leemos a partir de BP los elementos deseados.

De esta forma nos ahorramos tener que hacer múltiples PUSH y POP.

Si al direccionar, alguno de los registros implicados es BP, el micro autodetecta que queremos acceder a la zona de la Pila.

Si no ponemos BP, el segmento por defecto sería el especificado anteriormente (DS o SS). Ejemplo:

MOV AL , [BX−18h] [DS : BX−18h]AL

LODSB [DS : SI]AL ; SI++ ó SI −

MOV [DI],AX AX[DS : DI]

STOSB AL[ES : DI]

MOV AL,[BP+DI−428Ah] [SS : BP+DI−428Ah]AL

MOV BP,[384Ah] [DS : 384Ah]BP

Se usa DS porque BP no está dentro del direcionamiento

PUNTERO DE INSTRUCCIÓN:

El registro IP (Instruction Pointer) contiene el offset del contador de programa, es decir, cuando el procesador necesita una nueva instrucción de programa, la obtiene de la dirección indicada por el registro IP, que contiene el desplazamiento dentro del segmento apuntado por el registro CS desde donde se leerá dicha instrucción. Al igual que sucedía con el registro CS, éste sólo puede modificarse mediante las instrucciones de transferencia de control.

DE SEGMENTO:

Una posición de memoria queda determinada por una dirección de segmento y un desplazamiento relativo a dicho segmento denominado offset, donde el segmento se encuentra contenido en unos registros especiales denominados de segmento.

Cuando una instrucción hace referencia a una posición de memoria, ésta sólo especifica el offset de dicha posición, mientras que el segmento en que está contenida se obtiene de uno de los registros de segmento. Existen cuatro registros de este tipo, de 16 bits cada uno, que reciben el nombre de CS, DS, ES y DS.

Memoria

- CS = Code Segment Para código (instrucciones)
- DS = Data Segment Para Datos (constantes, variables,...) ***Código*** CS
- SS = Stack Segment Para la Pila
- ES = Extra Segment Para Datos Extra

No existe limitación alguna para tener varios segmentos de código, ***Datos*** DS

datos y pila, lo único necesario es cambiar la dirección de inicio de

los segmentos en el registro adecuado. La limitación es que en un

determinado instante, el micro sólo puede ver 4 segmentos ***Pila*** SS

(CS,DS,SS,ES), pues cada registro de segmento referencia

únicamente a un segmento de 64 KB. Así pues, podemos tener,

por ejemplo dos (o más) segmentos de código, y para acceder a

uno u otro, cambiaremos el valor del registro CS para que apunte al ***Datos Extra*** ES

comienzo del segmento deseado.

Registro CS (Code Segment): apunta siempre al segmento donde se encuentra la instrucción que se está ejecutando. Este registro no se puede modificar directamente y debe hacerse mediante las denominadas instrucciones de transferencia de control.

Registro DS (Data Segment): apunta al segmento de memoria donde las instrucciones leen los datos o los almacenan. Este registro puede cambiarse durante la ejecución del programa para así poder acceder a toda la memoria instalada en el sistema.

Registro ES (Extra Segment): indica una zona de memoria para datos adicional.

Registro SS (Stack Segment): apunta al segmento de memoria donde se encuentra la pila del procesador.

Cuando una instrucción necesita leer o escribir un dato en memoria, lo hace por defecto en el segmento apuntado por DS, aunque este funcionamiento puede variarse de forma que la instrucción obtenga el dato de cualquiera de los otros tres segmentos indicándolo de forma explícita en la instrucción, mediante los denominados prefijos de segmento, que veremos más adelante.

FLAGS:

El registro de banderas o flags (PSW, Program Status Word) se utiliza para indicar al programa ciertas condiciones que han tenido lugar como resultado de la última operación indicada. Cada uno de los bits del registro de banderas marca si se ha producido una de estas condiciones: 1 si se produjo o 0 si no tuvo lugar.

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

				OF	DF	IF	TF	SF	ZF		AF		PF		CF

Las banderas contenidas en este registro son las siguientes:

Flags Aritméticos:(0 , 2, 4, 6, 7 y 11):

CF (Carry Flag): Flag de acarreo en operaciones aritméticas. Toma el valor 1 si en la última operación de suma o de resta se produjo acarreo (o borrow en restas).

PF (Parity Flag): Flag de paridad. Toma el valor 1 cuando el resultado de la última operación tiene un número par de bits a uno.

AF Auxiliary Flag): Flag auxiliar. Es un tipo de acarreo especial que se utiliza para indicar un acarreo entre las cifras de un número en formato BCD.

ZF (Zero Flag): Flag de cero. Adopta el valor 1 en caso de que la última operación haya tenido un resultado igual a cero.

SF (Sign Flag): Flag de signo. Indica el signo de la última operación:

0 +

1 −

OF (Overflow Flag): Flag de desbordamiento. Cualquier operación que provoque un resultado que no quepa en el destino (desbordamiento) pondrá este indicador a 1.

Flags Software: (8, 9 y 10) (Modificables mediante instrucciones software):

DF (Direction Flag): Flag de dirección. Si tiene el valor 0, los punteros en instrucciones de proceso de cadenas se incrementan , y si tiene el valor 1 dichos punteros se irán decrementando.

IF (Interrupt Flag): Flag de interrupciones. Cuando se coloca el valor 0 en esta bandera, el procesador ignora las peticiones de interrupción que le hagan los dispositivos externos. Esto es útil para proteger secciones críticas del código que podrían dejar el sistema en estado inestable en caso de que se produjese una interrupción durante su ejecución.

TF (Trap Flag): Flag de paso a paso. Si éste flag tiene el valor 1, entonces el procesador genera una interrupción interna por cada instrucción que se ejecuta, posibilitando así a un programa de depuración el seguir el transcurso de un programa paso a paso.

PREFIJOS.

Un Prefijo es un byte que se coloca delante de una instrucción ensamblada (código máquina) y que modifica la ejecución de dicha instrucción. No todos los prefijos pueden ir con todas las instrucciones.

Instrucción ensamblada.

Byte de Prefijo

TIPOS DE PREFIJOS:

De Bloqueo LOCK Instrucción

SEG RegSeg Instrucción

De Segmento ó

RegSeg: delante del operando en memoria.

REP Instrucción.

REPE Instrucción.

De Repetición REPZ Instrucción.

REPNE Instrucción.

REPNZ Instrucción.

PREFIJO DE BLOQUEO.

Supongamos lo siguiente:

Proceso 1 Zona de memoria común.

Proceso 2

Para que un proceso no modifique los datos del otro proceso, se emplea una variable semáforo (es decir, que puede tomos dos valores). Si cada proceso accede a la tabla (memoria común) de la siguiente manera, aparentemente el acceso funciona (aunque, como veremos, no es así):

R V

Semáforo

Rojo Semáforo

Accesos

Verde Semáforo

El trozo de código dentro del óvalo debe ser una sola instrucción para que en caso de cambio de tarea, no quede a la mitad de su ejecución. Para ello se emplea la instrucción genérica Test&Set, que comprueba el semáforo y lo establece a un determinado valor.

Sin embargo este planteamiento puede dar problemas. El microprocesador asigna siempre una *Time Slice* (rodaja o porción de tiempo) a cada proceso (para que un mismo proceso no monopolice el uso del micro) para su ejecución. Cuando acaba el tiempo de un proceso, se guardan sus datos y se asigna el micro al siguiente proceso. Para saber cuándo acaba cada porción de tiempo, suele haber un *Timer* (temporizador) que envía una señal de interrupción cada x tiempo al micro advirtiéndole que el Time Slice del proceso actual ha acabado (es decir, el cambio de tareas se detecta/efectúa a través de una interrupción). Cuando se recibe la interrupción, se termina de ejecutar la instrucción en curso y justo después se atiende a la interrupción (la instrucción en curso no puede quedar a medio ejecutar).

En el 8086 no hay una instrucción específica de Test&Set, así que se emplea la instrucción XCHG, que realiza

algo parecido.

XCHG reg , memoria reg mem

mem reg

Para simular la instrucción Test&Set (que Intel decidió no incorporar al juego de instrucciones del 8086, la incorporó a partir del 386), en un 8086 habría que realizar el siguiente proceso:

Rojo reg

Micro 8086 Memoria

XCHG reg , semáforo reg

(memoria)

R V

reg ? reg. aux

Accesos

Verde Semáforo

Supongamos ahora que tenemos dos o más CPUs 8086 con una cola de procesos. Una vez que un micro coge los buses, no los suelta hasta que acaba de ejecutar la/las instrucciones (las que dé tiempo en el Time Slice). Lo que hace <LOCK Instrucción> (nótese que LOCK va con una sóla instrucción) es precisamente blindar el proceso, para que no se permita cambiar de tarea hasta que la instrucción en curso no se haya terminado de ejecutar, impidiendo al otro micro el meterse por medio cuando el micro actual está ejecutando y tiene los buses bajo su control.

LOCK es una instrucción que se utiliza en aplicaciones de recursos compartidos para asegurar que no accede simultáneamente a la memoria más de un procesador. Cuando una instrucción va precedida de LOCK, el procesador bloquea automáticamente el bus, introduciendo una señal por la patilla LOCK.

La sintaxis de la instrucción Time&Set simulada es por tanto:

LOCK XCHG reg, semáforo

Cuando el procesador encuentra esa instrucción, pide los buses pero con bloqueo (el bloqueo del bus de acceso a memoria no desaparece hasta que se acaba de ejecutar la instrucción a la que modifica LOCK). Así se asegura que sólo un procesador acceda a la memoria en un determinado momento.

PREFIJOS DE SEGMENTO.

Indican al ensamblador que la dirección física de algún operando debe calcularse con algún registro de segmento (RegSeg) distinto al que se usa por defecto. Hay dos formas de indicarlo:

Si los operandos son implícitos:

Se coloca delante de la instrucción **SEG RegSeg**

Ej:

- SEG SS MOVSB

[SS : SI] [ES : DI] ; SI ± ±; DI ± ±

- SEG DS STOSB

AL ES : DI; DI ± ± Incorrecta

Si los operandos son explícitos:

Se coloca delante del operando en memoria r**egseg :**

Ej:

- MOV AL, ES : [BP+DI−18h]

[ES : BP+DI−18h] AL

- MOV CS : [BX+1248h], AL

AL [CS : BX+1248h]

PREFIJOS DE REPETICIÓN.

Pueden usarse solamente en **instrucciones de manipulación de Strings.**

Las instrucciones de manipulación de strings son las siguientes:

LODSB [DS : SI]8 AL ; SI ±±

LODSW [DS : SI]16 AL ; SI ±2 SI

De Transferencia STOSB AL [ES : DI]8 ; DI ±±

(Prefijo REP) STOSW AX [ES : DI]16 ; DI ±2 DI

MOVSB [DS : SI]8 [ES : DI]8 ; SI±± ; DI ±±

MOVSW [DS : SI]16 [ES : DI]16 ; SI± 2 SI ; DI± 2 DI

SCASB AL − [ES : DI]8 Act.Flags ; DI ±±

De Comparación SCASW AX − [ES : DI]16 Act.Flags ; DI ±2 DI

(Prefijos) CMPSB [DS : SI]8 − [ES : DI]8 Act.Flags ; SI±± ; DI ±±

(REPE o REPZ) CMPSW [DS : SI]16− [ES : DI]16 Act.Flags ; SI±2 SI;DI ±2DI

(REPNE o REPNZ)

REP

N S

CX=0

Ejecuta instrucción

CX − − (no afecta a flags)

REPE / REPZ REPNE / REPNZ

N S N S

CX=0 CX=0

Ejecuta instrucción Ejecuta instrucción

CX − − (no afecta a flags) CX − − (no afecta a flags)

=1 =0 =0 =1

Fz Fz

INTERRUPCIONES Y EXCEPCIONES.

La Tabla Vectorial de Interrupciones es una tabla de 256 entradas de 4 bytes cada una (256 punteros), por lo que su tamaño es de 1KB = 400H, situada justo al comienzo de la memoria del ordenador (0000h:0000h). Toda interrupción que se ejecute supone una ruptura con salto intersegmento (modifica el registro de segmento).

Tipos de interrupciones: Interrupciones Hardware, Software, No Enmascarables y Excepciones.

Interrupciones Hardware:

Algunas veces, un dispositivo necesita llamar la atención del procesador para indicarle, por ejemplo, que tiene un dato disponible o que ha terminado de realizar la acción que se le había encargado .Este es el sistema utilizado, por ejemplo, por el teclado, el cual envía una interrupción al procesador cada vez que se pulsa una tecla, o también el utilizado por la unidad de disco para señalar que ha terminado de escribir los datos en el disquete. A este tipo de interrupciones también se les denomina externas, para diferenciarlas de las de tipo interno generadas por el propio procesador para indicar errores de tipo matemático o como ayuda a la depuración de programas.

Interface

IRQ0

INT

PIC Interface

8086 INTA IRQ1

. . . . . .

IRQ7

Interface

Buses

Vector de interrupción (número)

Dado que el microprocesador es millones de veces más rápido que los periféricos, no es útil que esté preguntando a los periféricos y quede esperando respuesta (pérdida de tiempo). La misión del PIC (controlador de interrupciones) es multiplexar las señales de interrupción que vienen de los periféricos.

Las interrupciones pueden ser enmascarables en función del flag de interrupción (Si FI = 1 no se atienden).

El proceso de detección y atención a la interrupción es el siguiente:

- Cuando llega una señal de interrupción al PIC, este le manda una señal INT (de petición de interrupción) al micro.
- El micro finaliza la ejecución de la instrucción en curso.
- Analiza el flag de interrupción:
- Si es 1, continúa sin atender la señal (sigue atendiendo las instrucciones anteriores).
- Si es 0, atiende a la interrupción y activa la señal INTA (INTerrupt Acknowledge).
- El PIC detecta esta INTA y envía al DBUS el vector de interrupción (num. de 8 bits sin signo).
- El micro recoge el vector del DBUS. {Ciclo de interrupción}
- Se meten en la pila el PSW, CS e IP.
- Se multiplica por 4 el vector y se lee de esa posición en la tabla de vectores.

Si hay varias IRQ's pidiendo INT, el PIC elige, por un sistema de prioridades, que IRQ atender, pudiendo enmascarar una a una cualquiera de las IRQ's.

El vector puede ser cualquiera entre 05h y FFh (inclusive).

Interrupciones Software:

Para poder acceder a diversas rutinas (rutinas DOS, BIOS o de usuario), se accede a través de interrupciones software. Para ello, se colocan en unos registros determinados los parámetros que necesita la subrutina y se llama a la interrupción correspondiente (mediante la instrucción INT vector). Una vez completado el proceso, la interrupción retornará devolviendo resultados en registros. Además, en los casos en que una misma interrupción ofrece varias rutinas, en el registro AH contiene un número identificativo de la función que se desea utilizar.

Interrupciones No Enmascarables (NMI):

- Son interrupciones hardware.
- No les afecta el FI, por lo que se atienden siempre.
- La señal de control es distinta NMI
- En el caso del 8086 el único vector válido es el 02.
- Suelen estar asociadas a interrupciones terminales.

Excepciones:

Son interrupciones que genera el propio micro durante la ejecución del programa cuando se produce una anomalía (por ejemplo un error aritmético, como división entre cero, que el micro no sabe tratar). Cada anomalía posible tiene asignado un determinado número de interrupción (nº de Excepción), y viene dado por el fabricante del procesador. Por ejemplo, en el 8086, la división entre cero provocará la Excepción 00h).

¿Cómo se atienden?

Lo que nos ofrecen cualquiera de los tipos de interrupción es un vector (nº de 8 bits sin signo) a través del Bus de Datos. La forma de atender ese vector (que se obtendrá de una forma u otra dependiendo del tipo de interrupción) es la misma en cualquier caso. El proceso de atención a la interrupción es el siguiente:

- Se guarda en la Pila (automáticamente) el PSW (Flags), el CS y el IP:

SP − 2 SP ; PSW [SP]

SP − 2 SP ; CS [SP]

SP − 2 SP ; IP [SP]

Antes Tras guardarlos en la Pila

SP IP

SS SS SP CS

SP PSW

SP SP

- Se ponen a 0 los flags I y T. (0 FI ; 0 FT)

Flag I: Cuando va a atender a una interrupción, el micro inhabilita las interrupciones para que si llega otra interrupción mientras atiende una, no haga caso y termine de atender a la que esté en curso.

Rutina Atención Interrupción

Deshabilitar ints. Habilitar interrupciones

INT

8086

Recupera

PSW anterior

Flag T: Supongamos que tenemos un depurador y vamos a depurar un programa. Si entre medias surge una interrupción y FT=1, se trataría la rutina de atención a la interrupción también de instrucción en instrucción (paso a paso), cosa no deseada.

- Se lee de la Tabla Vectorial de Interrupciones la dirección inicial de la rutina de atención y se salta a ella (las direcciones de las rutinas de atención a interrupciones de almacenan en dicha tabla, a la que debemos

acceder, conocido el vector, para determinar dónde se encuentra la rutina de interrupción deseada). La dirección de la rutina de atención a la interrupción, CS:IP, se obtiene:

(4 * Vector) CS

(4 * Vector + 2) IP

Vector 0 0 1 2 3

Vector 1 4 5 6 7

... ...

4v 4v+1 4v+2 4v+3

Vector V 1 A 8 2 4 F 3 C

... ...

3F8 3F9 3FA 3FB

Vector FEh

3FC 3FD 3FE 3FF

Vector FFh

Menos singnificativa. Más significativa. Menos singnificativa. Más significativa.

Offset Segment

Por ejemplo, la rutina que atiende al vector V empieza en la dirección: 3 C 4 F h **:** 8 2 1 A h

VECTORES RESERVADOS.

En el 8086 los vectores reservados van del 0 al 4:

Vector 0 Desbordamiento en instrucciones de división. (Excepción)

Vector 1 Interrupción por finalización de instrucción si FT = 1.

Vector 2 Interrupción hardware no enmascarable (señal NMI).

Vector 3 Interrupción software especialmente diseñada para implementar Breakpoints en

depuradores, ya que la INT 3 ocupa 1 byte (todas las demás INT ocupan 2 bytes).

Vector 4 Producida por una INT 0 si el Fo = 1.

División:

Divid16 / Divid8 se espera que produzca: Cociente8 y Resto8

1111 1111 1111 1111 / 0000 0001 Desbordamiento de división INT 0

Cuando FT = 1, se producirá una interrupción cada vez que se termine de ejecutar una instrucción.

La mayoría de las INT hardware van por el sistema conocido: diálogo entre Microprocesador − PIC.

INT

8086 PIC

INTA

Vector

Sin embargo, también hay otra posibilidad, que es cuando sucede una INT de tipo NMI. Ésta interrupción tiene una línea (cable) asignada, y cuando se activa, no hay diálogo entre Microprocesador − PIC, sino que directamente la NMI tiene siempre asignado el Vector 2 (INT 2):

8086 PIC

NMI (Vector 2)

La interrupción 3 (INT 3) tiene un código de operación (OPCODE) de 1 byte, mientras que las demás tienen 2 bytes.

Un Breakpoint (punto de ruptura) indica el punto en que el programa dejará de ejecutarse normalmente y pasará a hacerlo paso a paso (para que pueda usarlo un depurador).

El depurador debe estar diseñado de la siguiente forma:

en donde se sitúa el Breakpoint, se coge la instrucción, (que se almacenará en el depurador), y la sustituye por una llamada a la INT 3 (por lo que dicha llamada debe ocupar 1 byte, ya que la instrucción sustituida puede ser de las que ocupan un solo byte).

INT 0 mira el Flag de Overflow, y si está a 1 hay overflow, lo que provocará una INT 4, saltando a la rutina de atención a la INT 4 que atenderá dicho desbordamiento.

Oper ...

INT 0 JNO xxxx

.... JMP/CALL rutina

ope xxxx:

INT 0 ...

VECTORES DE INTERRUPCIÓN.

Los siguientes vectores de interrupción son los mas usados:

Nº Vect. Función que realiza

0−4 Reservados para el micro

5 (BIOS, Soft.) Volcado de pantalla en impresora al pulsar IMPR PANT

8−0FH (Hard.) 1er controlador de interrupciones

8 Periódica cada 1/18.2 segundos

9 Teclado

0BH COM1

0CH COM2

0DH LPT2 (en XT disco duro)

0EH Diskette

0FH LPT1

10H−1AH Servicios BIOS

10H Vídeo

13H Discos y Floppy

14H Comunicaciones serie

16H Teclado

17H Impresora

1AH Fecha y Hora

1BH (Soft.,BIOS) Tratamiento CTRL+PAUSE

20H Terminación del programa

21H Servicios del SO

22H Terminación normal

23H Terminación por aborto

24H Terminación por error critico

25H−26H Lectura y escritura en discos y floppys por número de sector

27H Terminación quedando residente

33H Ratón (driver del ratón) (Soft.)

40H−47H −−−−−−−−−−−−−−disponibles−−−−−−−−−−−−−−−−−−

50H−57H (Hard.) aquí se tratan los vectores 08H a 0FH cuando se pasa a Modo Protegido

58H−6FH −−−−−−−−−−−−−−disponibles−−−−−−−−−−−−−−−−−−

70H−77H(Hard.) Segundo controlador interrupciones

70H Reloj tiempo real

74H Ratón (Hard.)

75H Coprocesador

76H Disco duro

78H−7FH −−−−−−−−−−−−−−disponibles−−−−−−−−−−−−−−−−−−

80H−FFH −−−−−−−−−−−−−−disponibles−−−−−−−−−−−−−−−−−−− (Antiguamente BASIC de la PROM)

EL PREFETCH.

Lectura anticipada de código.

El 8086 se puede considerar internamente dividido en 2 partes: BIU (Bus Interface Unit) y EU (Execution Unit).

8086

BIU EU

BUSES

Pila de Prefetch.

(FIFO)

El BIU es la unidad que se encarga de la comunicación del micro con los buses externos al microprocesador. El EU se encarga de interpretar y ejecutar las instrucciones.

Cuando el EU necesita un dato, se lo debe pedir al BIU, el cual gestionará dicha petición. Cuando el BIU no tiene nada que hacer, lee las siguientes instrucciones a ejecutar y las almacena en la Pila de Prefetch (estructura FIFO), que es una pila interna del micro, de 6 bytes y no modificable por nosotros (es gestionada interiormente por el micro).

Esto es útil sólo en el caso de que las siguientes instrucciones leídas sean las que se vayan a ejecutar, pero si el

EU se encuentra con una instrucción de salto, entonces las instrucciones introducidas en la Pila de Prefetch no nos sirven, por lo que el EU vaciará dicha pila (además de actualizar valores de IP y demás registros).

CS : IP JMP xxxx

xxxx:

Hay casos (código automodificable) que dificultan el antidebug, en los cuales al leer instrucciones y depositarlas en la Pila de Prefetch, si luego el código se automodifica, lo haría en memoria, pero no en la Pila de Prefetch. Para ignorar las instrucciones de la Pila de Prefetch, se puede, p.e., hacer un salto (JMP) a la siguiente instrucción, lo que a simple vista es poco útil, pero EU detecta dicho salto, y borra la Pila de Prefetch (el salto provoca el borrado de la Pila de Prefetch).

JMP xxxx Provoca el borrado de la Pila de Prefetch.

xxxx:

OTRAS CARACTERÍSTICAS.

Ciclo de máquina típico: 4 ciclos de reloj.

Código de operación: 1 o 2 bytes.

Tamaño de la instrucción: 1 a 6 bytes.

Utiliza DMA: El micro pasa al estado de parada (HALT) del que sale por cualquier interrupción HLT.

OPERANDOS.

Memoria

Micro 8086

[DS:SI]8

Byte

DS

AH AL [DS:SI]16

AX

(Word)

[ES:DI]8

ES

[ES:DI]16

Load

mem reg

Store

reg mem

ACCESOS A MEMORIA.

Tipo de Ciclo Máquina (acceso)			**Segmento**	**Offset**	**¿Puede usarse otro segmento?**	**Comentarios**	
Fetch: Lectura de instrucción.			CS	IP	NO	CS:IP hace las veces de contador de programa. En cada instrucción leída se incrementa IP. Si IP sale del segmento CS por abajo, vuelve a 0000h de CS.	
Ciclos automáticos de Pila.			SS	SP	NO	SS:SP hace las veces de puntero de pila. Los incrementos/decrementos sólo se producirán sobre SP.	
Datos	Inst. de Manip. Strings.	Ptro. destino.	**ES**	DI	NO		
				Ptro. Fuente.	DS	SI	SI
			Otras instruccs.	BP interviene en inst.	SS	Varios modos de direcc.	SI (Prefijo de segmento)
				BP no interviene en inst.	DS	Varios modos de direcc.	SI (Prefijo de segmento)

- **<u>Ciclos automáticos de Pila:</u>**

- Introducen o sacan algo de la Pila. Por lo tanto, siempre están referidos al Puntero de Pila y lo modifican.
- En la instrucción no se indica explícitamente a qué dirección de la Pila se accede.
- Los hay en las instrucciones:

PUSH, POP

CALL, INT, INTO

RET, IRET

- El incremento o decremento depende de:

BYTE: Se incrementa /decrementa el puntero en 1 posición.

Si el elemento es un

WORD: Se incrementa/decrementa el puntero en 2 posiciones.

- SP es el offset del último byte ocupado de la pila, por lo que el par SS:SP apunta a la Cima de la Pila.
- Si seguimos intentando llenar la Pila al llegar SP a 0000h (recordemos que la pila crece hacia abajo en posiciones de memoria), el puntero SP volvería cíclicamente al FFFFh de SS (Segmento de Pila).

IP

CS **Código** (1segmento = 64KB max)

SS **Pila** (1segmento = 64KB max)

SP Parte ocupada

de la Pila

- **Instrucciones de Manipulación de Strings.**

- Dejan incrementados o decrementados los punteros que intervengan para facilitar la realización de bucles y repeticiones.
- Sintácticamente no tienen operandos y los mnemónicos de estas instrucciones acaban en SB o en SW. Concretamente son:

LODSB, LODSW

STOSB, STOSW

MOVSB, MOVSW

SCASB, SCASW

CMPSB, CMPSW

Instrucción STOS / STOSB / STOSW: (almacena cadena)

Sintaxis: STOS cadena_destino

STOSB (bytes)

STOSW (words)

Transfiere el operando origen almacenado en AX (si es word) o en AL (si es byte), al destino direccionado por ES : DI. Tras la operación, DI se incrementa o decrementa según el flag DF para apuntar el siguiente elemento de la cadena. cadena_destino es un operando redundante que sólo indica el tipo de dato (byte o palabra) a cargar, es más cómodo colocar STOSB o STOSW para indicar bytes/words.

Instrucción MOVS/ MOVSB / MOVSW: (copia cadena)

Sintaxis: MOVS cadena_destino, cadena_origen

MOVSB (bytes)

MOVSW (words)

Transfiere un byte o palabra de la cadena origen direccionada por DS : SI a la cadena destino direccionada por ES : DI, incrementando o decrementando a continuación los registros SI y DI según el valor del flag DF en una o dos unidades, dependiendo de si se trabaja con bytes o con palabras. Si se indica un registro de segmento, éste se sustituirá en la cadena origen al DS ordinario.

A base de repeticiones se puede copiar una zona de memoria a otra (para lo que precisaríamos un contador para saber qué número de elementos (bytes con MOVSB, o words con MOVSW) se copian.

DS

SI

DI

ES

Hay algunas instrucciones de manipulación de string que trabajan con uno otro puntero o incluso los dos:

PUNTERO FUENTE: DS : SI (LODSB y LODSW)

PUNTERO DESTINO: ES : DI (STOSB y STOSW)

(SCASB y SCASW)

CON LOS DOS PTROS: (MOVSB y MOVSW)

(CMPSB y CMPSW)

- **Otras instrucciones.**

- Normalmente el operando es explícito.
- El contenido de memoria se expresa con corchetes [ ] en vez de con paréntesis.
- Lo que hay dentro de los paréntesis siempre es el Offset.
- Existen las siguientes formas de expresarlo:

[Offset] Ej: MOV A , [4832h]

[Reg] Ej: MOV [BC] , A

[Seg + Desp] Ej: MOV

[Reg + Reg] Ej: MOV [BC + X] , A

[Reg + Reg + Desp] Ej: MOV

PUERTOS.

BHE

A0