Aunque actualmente existen ordenadores con diferentes *CPU's*, estudiaremos el microprocesador 8086, ya que fue el utilizado en el primer *PC*. El porqué de esta elección, es debido a que este microprocesador va a imponer la forma de operar de los siguientes (para mantener la compatibilidad entre ellos).

El 8086 aparece en el mercado en 1978 y es uno de los primeros microprocesadores de 16 *bits*. Su bus de direcciones de 20 líneas proporciona un margen de direccionamiento de 1 *Mbyte*. Para compatibilizarlo con el microprocesador 8085 (estaba muy difundido por aquel entonces) de 8 *bits*, *Intel* desarrolló el 8088 que utiliza 16 *bits* internamente pero que externamente tiene un bus de 8 *bits*. Esta característica provoca que sea más lento, dado que los datos de 16 *bits* deben ser accedidos en dos ciclos. Pero, los programas escritos para el 8086 funcionaban en el 8088 y viceversa.

A continuación, se describen algunas características de este microprocesador:

- Los buses de datos y direcciones están multiplexados. Esto es, se extraen los datos y las direcciones por las mismas patillas del microprocesador (*AD0...AD15*, donde la *A* se refiere a *Address* y la *D* a *Data*). Las patillas de las primeras 16 líneas del bus de dirección son también 16 líneas de datos. El porqué de esta doble función es muy simple: cuando se diseñó el 8086, las técnicas de encapsulado obligaban a reducir lo máximo posible el número de terminales, por lo que se decantó por esta técnica.
- Las líneas de dirección de la *A16...A19* también están multiplexadas, suministrando información de control cuando la dirección está validada como muestra la Figura 6.8.

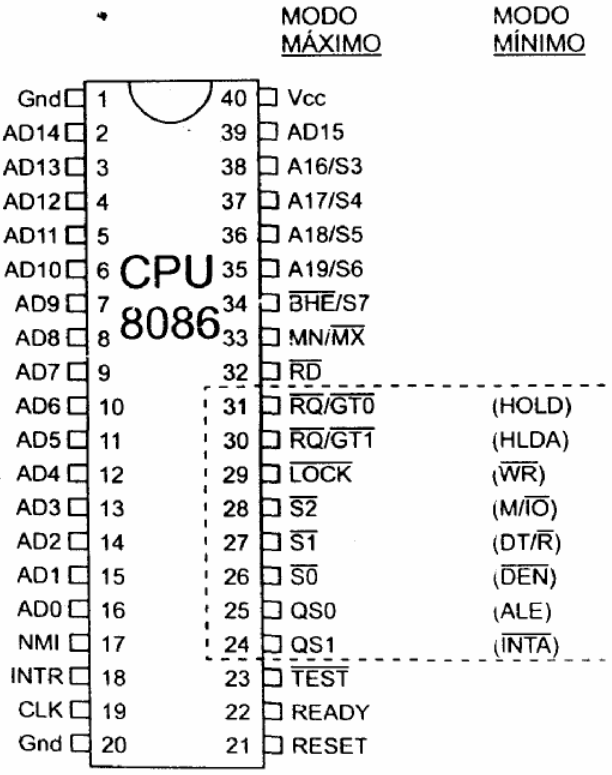


**Figura 6.8. Patillaje del microprocesador 8086.**

- Este microprocesador, tiene dos modos de trabajo:
  - *Modo MÁXIMO*, cuando trabaja con otros microprocesadores alternándose el control de los buses.
  - *Modo MÍNIMO*, en este modo de funcionamiento, el microprocesador trabaja sólo con los chip auxiliares de la familia.

## 6.5.1. Arquitectura interna

La estructura interna del 8086 está formada por dos partes o unidades de proceso como muestra la Figura 6.9. Esta arquitectura, vino impuesta para mantener la compatibilidad con el 8085 (microprocesador de *Intel* de 8 *bits*).

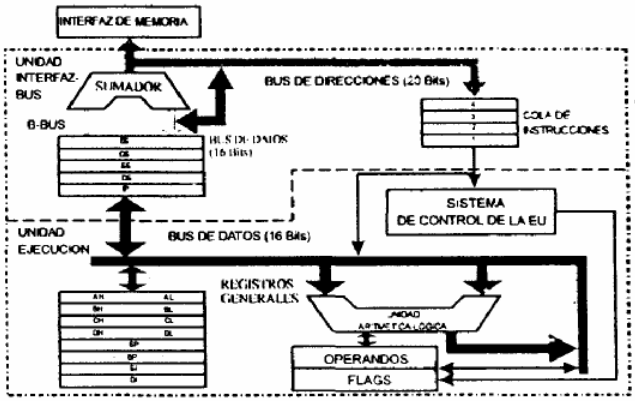


**Figura 6.9. Arquitectura interna del microprocesador 8086**

A continuación, pasamos a describir cada una de estas partes:

## 6.5.2. EU (Execution Unit o Unidad de Ejecución)

Es donde tiene lugar el procesamiento de los datos dentro del 8086, es la encargada de interpretar y ejecutar las instrucciones. El margen de memoria que maneja la *EU* es de 64 *Kbytes* y sus buses internos son de 16 *bits*.

El juego de registros de la *EU* es parecido al del 8085 y constituye la base para su compatibilidad. Todos los registros del 8085 tienen una correspondencia directa con algunos del 8086.

La *EU* recoge las instrucciones que han sido buscadas por la *BIU*, y las procesa. Una vez procesadas, devuelve a la *BIU* las direcciones donde se encuentran los operandos. Una vez recibidos los operandos, los procesa y el proceso termina cuando la *EU* devuelve a la *BIU* los resultados, para almacenarlos en memoria.

La Unidad de Ejecución, consta de los siguientes elementos:

- ***ALU* de 16 *bits*:** La *ALU* es una Unidad Aritmética/Lógica que se encarga de realizar las operaciones lógicas (operaciones a nivel de *bit*) y aritméticas como la suma, resta...

- **Registro de Estado:** Contiene los *flag* de estado de la *CPU*, estos *flags* nos indican en cada momento la situación (estado) del microprocesador.
- **8 registros de 16 *bits*:** Estos registros, se utilizan para manipular datos, almacenar resultados intermedios y seguir el estado de la pila (*pushdown*). Estos registros pueden dividirse en:

  a.- *4 registros generales:* Conocidos con el sobrenombre de grupo *HL*, porque pueden ser subdivididos en *bytes* altos y bajos (Figura 6.10), convirtiendose en 8 registros de 8 *bits* cada uno. Estos registros generales se utilizan para la manipulación de datos y son:

  * ***AX (AH y AL):*** Acumulador.
  * ***BX (BH y BL):*** Registro base, se utiliza como registro base para los direccionamientos.
  * ***CX (CH y CL):*** Registro contador, se utiliza como contador de bucles con la instrucción *LOOP*, en rotaciones y en desplazamientos.
  * ***DX (DH y DL):*** Registro de datos, se utiliza para el almacenamiento de datos y se utiliza conjuntamente con el *AX* (Acumulador) en las operaciones de multiplicación y división.

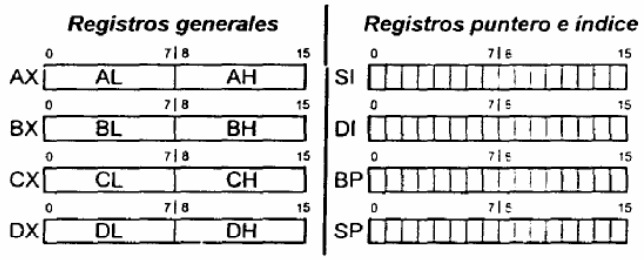


**Figura 6.10. Registros generales del 8086.**

  b.- *4 registros puntero e índice*: Al contrario que los registros generales, no se pueden dividir en *byte* alto y bajo y se utilizan para general el *offset* de la dirección de un segmento de memoria (no se preocupe sino entiende alguno de los términos tales como *offset*, segmento..., ya que serán estudiados al detalle más adelante.). Estos registros manejan direcciones de 16 *bits*, de aquí que el rango de direcciones que maneja sea de 64 *Kbytes* y por ello necesite de la *BIU* para generar la dirección física (la que sale por las patillas del microprocesador). También pueden participar junto con el grupo de registros generales en las operaciones aritméticas y lógicas del 8086. Estos registro son:

  * ***SP* (*Stack Pointer* o puntero de pila):** Apunta a la zona de memoria de la pila. Funciona automáticamente en todos los accesos a la pila como son las llamadas a interrupción, subrutinas...
  * ***BP* (*Base Pointer* o puntero base):** Junto con el segmento de la pila, permite acceder a los datos de la pila.
  * ***SI* (*Source Index* o registro indice de destino):** Se utiliza en el manejo de cadenas (*string*) como fuente se los datos originales. Si está libre puede utilizarse para cualquier fin.
  * ***DI* (*Destination Index* o registro índice fuente):** También se utiliza en el manejo de cadenas (*String*) como la tabla o cadena destino de los datos resultantes.

Existe otro registro puntero, pero no está ubicado en la *EU* sino en la *BIU*. Se trata del *IP* o *puntero de instrucciones*, que realiza las veces de contador de programa almacenando la dirección de la próxima instrucción a partir del comienzo del segmento de código (*CS*).

## 6.5.3. BIU (Bus Interface Unit o Unidad de Interface del BUS)

Es la que se encarga de generar las direcciones físicas de 20 *bits* y asegura que el bus se está utilizando a toda su capacidad. Para acelerar el proceso, la *BIU* busca las instrucciones y las direcciones de almacenamiento y lectura antes de que sean necesitadas por la *EU*, y las almacena en la cola de instrucciones. Dejando así a la *EU* libre para concentrarse en el tratamiento de datos y la ejecución de instrucciones.

Esta unidad está compuesta por los siguientes elementos:

- **Registros de segmentos:** Son 4 registros de 16 *bits* cada uno, se utilizan en el mecanismo de direccionamiento de la memoria. Cada uno de estos registros, guardará la dirección de un segmento. A continuación, se describen cada uno de estos registros:
  - ***CS* (*Code Segment* o Segmento de Código):** Señala el segmento donde se encuentran las instrucciones del programa que se está ejecutando. Se utiliza en combinación con el puntero de instrucciones (*IP*).
  - ***DS* (*Data Segment* o Segmento de Datos):** Señala el segmento que contiene los datos que maneja el programa que se está ejecutando. Se utiliza en combinación con los registros *BX*, *SI* o *DI*.
  - ***SS* (*Stack Segment* o Segmento del Stack):** Señala el segmento donde se va a encontrar la pila. Se utiliza en combinación con los registros *BP* y *SP*.

    a.- *BP* es automático, cuando llega una interrupción.

    b.- *SP* es manual, y su valor viene condicionado por las instrucciones *PUSH* (guardar un dato en la pila) y *POP* (sacar un dato de la pila).
  - ***ES* (*Extra Segment* o Segmento Extra):** Señala la dirección de un segmento adicional. Este segmento se puede utilizar para diferentes funciones, entre ellas, manejo de cadenas, aunque también se suele utilizar para datos en combinación con *BX*, *SI* o *DI*.

En la Tabla 6.6, se muestra qué registro indica el desplazamiento dentro de cada segmento, así como su función.

| Segmento | Offset | FUNCIÓN |
|---|---|---|
| CS | IP | Aquí se encuentran las instrucciones de programa. |
| SS | SP y BP | Es donde se va ha encontrar la pila. |
| DS | SI, DI y BX | Aquí se encuentran los datos del programa. |
| ES | SI, DI y BX | Cadenas / Datos. |

- **Puntero de Instrucciones (*IP*):** Hace las veces de contador de programa. Se incrementa cada vez que se ejecuta una instrucción, para saber cual es la próxima instrucción a ejecutar.

- **Cola de Instrucciones:** Es un registro formado por una memoria tipo *FIFO* (*First Input, First Output*, es decir, el primer *byte* que se almacena, luego es el primero en salir) de 6 *bytes*, aunque en el 8088 es de solo 4 *bytes*. Se utiliza para que la *BIU*, deposite aquí las instrucciones contenidas en la memoria antes de que se vallan a ejecutar, y la *EU* las recupere de la cola en lugar de la memoria, lo que se traduce en un mejor rendimiento. La Figura 6.11 muestra el funcionamiento de la cola de instrucciones.

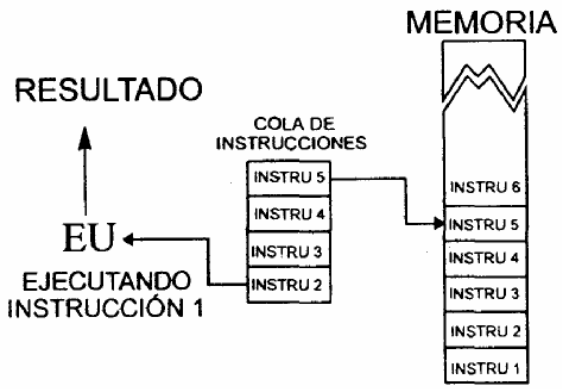


**Figura 6.11. Funcionamiento de la cola de instrucciones.**

- *ALU*: Se utiliza para realizar el desplazamiento de cuatro *bits* a la izquierda del segmento base, al que luego se le suma el *offset* como explicaremos en el siguiente apartado.

## 6.5.4. El direccionamiento de la memoria

Una vez vistos todos los tipos de registros de segmento que tiene el 8086, vamos a ver el funcionamiento de la memoria, quizás el tema más espinoso de la familia 8XX86 de *Intel*. Todo este asunto comenzó con el microprocesador de 8 *bits* 8085 de *Intel* que tenía en el mercado. Este microprocesador, como todos los de 8 *bits* podía direccionar hasta 64 *Kbytes* de memoria con sus 16 líneas de direcciones y sus registros de 16 *bits*.

*Intel*, se propuso sacar al mercado un nuevo microprocesador de 16 *bits* que pudiera manejar más memoria (1 *Mbyte* con 20 líneas de direcciones). Pero quería mantener la compatibilidad con el 8085 (muy extendido por aquel entonces). Es decir, que los programas escritos para el microprocesador de 8 *bits* pudieran funcionar en el de 16 *bits*. Para que la compatibilidad fuera posible, era necesario que los registros internos siguieran teniendo 16 *bits* de direcciones. Pero una *CPU* de 16 *bits* podía solamente acceder a 65.536 lugares diferentes (64 *Kbytes*) en memoria, e *Intel* pretendía ampliar ese rango a 1 *Mbytes*. Por lo que se ideó un sistema llamado *segmentación*.

## 6.5.5. Segmentación de la memoria

Para mantener la compatibilidad, la Unidad de Ejecución trabajaría como en el 8085 con 16 *bits* por lo que sólo tendría acceso a un segmento (en la terminología del 8086 se define un *segmento* como un espacio de posiciones de memoria consecutivas de 64 *Kbytes*). Por lo que la *BIU* sería la que se encargaría generar la dirección de 20 *bits* ayudada por un *desplazamiento* (*offset*). Esta interrelación entre el segmento y el desplazamiento se muestra en la Figura 6.12.

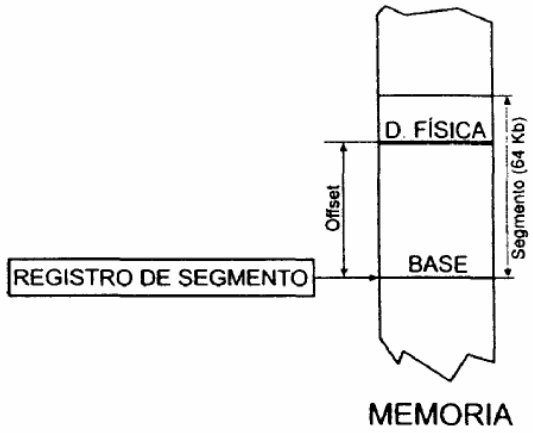


**Figura 6.12. Memoria segmentada.**

En consecuencia, vamos a tener la memoria dividida en segmentos, que funcionan igual que en el 8085. Para acceder a una posición de memoria determinada, elegimos uno de estos segmentos de 64 *Kbytes* mediante un registro de segmento (**base**). Este segmento, puede encontrarse en cualquier lugar del espacio de 1 *Mbyte*. Una vez seleccionado el segmento, elegiríamos la posición dentro de ese segmento mediante el desplazamiento (*offset*).

La *BIU* es la encargada de generar las direcciones físicas de 20 *bits* a partir de la base y el *offset*. Como hemos estudiado, la base del segmento la genera la *EU* y como tiene solamente 16 *bits*, para convertirla en una dirección física de 20 *bits*, la multiplica por 16. Recordemos que para multiplicar por 16 en binario, se añaden 4 ceros por la derecha.

Por ejemplo, si la *EU* genera la dirección de segmento A000H, ésta corresponde al segmento que comienza en la marca de 640 *Kbytes*. Porque A000H es igual a 40.960, que cuando se multiplica por 16 obtenemos A0000H que es igual 65.5360.

Ya tenemos seleccionado el segmento, ahora hay que especificar en qué lugar dentro de éste se encuentra el dato deseado. Para ello, utilizamos el desplazamiento (*offset*), que indicará la dirección de memoria a la que se dirige (dirección efectiva o física).

**DIRECCIÓN FÍSICA (ABSOLUTA)= (BASE · 16) + *OFFSET***

Por lo que una dirección podría indicarse como: A000H:1000H, donde A000H representa el segmento (*CS*) y 1000H el desplazamiento desde la base del segmento (*IP*). La Figura 6.13, muestra gráficamente su obtención por medio de la *ALU*.

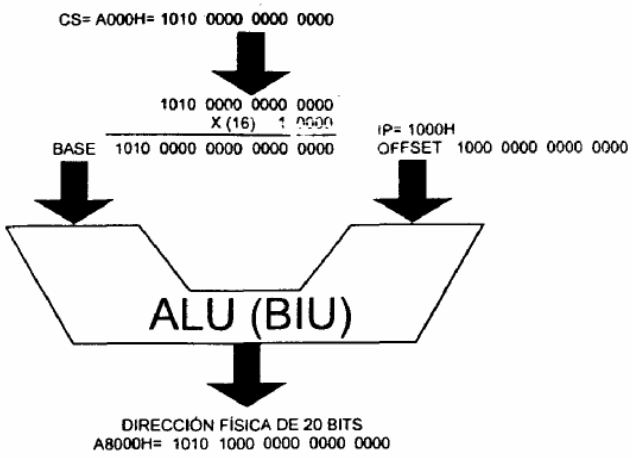


**Figura 6.13. Obtención de la dirección efectiva.**

También puede suceder, que nos den la posición de memoria en términos absolutos. Es decir, en un único número referido a cero. Por ejemplo, si nos dicen que la posición de memoria es la 1F800H. Pueden definirse tantos segmentos como se desee, con la única limitación de que debe comenzar en una dirección múltiplo de 16 (0, 16, 32, 64...). Podrían definirse, por tanto, hasta 65.536 segmentos diferentes.

La base del segmento se encuentra contenida en un registro específico denominado *registro de segmento*, cuyo tamaño es de 16 *bits*. El *offset*, también de 16 *bits*, y es generado de diferentes formas, según el tipo de acceso realizado y el modo de direccionamiento empleado.

A continuación, vamos a realizar varios ejemplos para comprender mejor este mecanismo de segmentación.

- Si tenemos un segmento= 4600H y un desplazamiento (*offset*)= 1800H, la dirección física o absoluta a la que accederá el microprocesador será la siguiente:

| | | |
|---|---|---|
| SEGMENTO | 4 6 0 0 | ⇨ desplazado 4 *bits* |
| DESPLAZAMIENTO | + 1 8 0 0 | |
| DIRECCIÓN ABSOLUTA | 4 7 8 0 0 | ⇨ dir. física de 20 *bits* (0100 0111 1000 0000 0000) |

- Si tenemos un segmento= 3950H y un desplazamiento (*offset*)= E300H, la dirección física o absoluta a la que accede el microprocesador es la siguiente:

| | | |
|---|---|---|
| SEGMENTO | 3 9 5 0 | ⇨ desplazado 4 *bits* |
| DESPLAZAMIENTO | + E 3 0 0 | |
| DIRECCIÓN ABSOLUTA | 4 7 8 0 0 | ⇨ dir. física, que se envía por el bus de direcciones |

Como hemos podido comprobar, una misma posición de memoria puede pertenecer a varios segmentos distintos, (hasta a 4096 segmentos), como muestra la Figura 6.14. Por ejemplo, la dirección física 0B200H puede ser referida, entre otras, de las formas mostradas en la Tabla 6.7.

| BASE | OFFSET |
|---|---|
| 0B00H | 0200H |
| 0020H | B000H |
| 0B20H | 0000H |

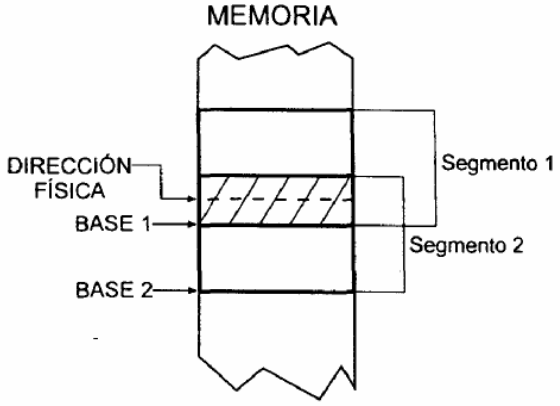


**Figura 6.14. Solapamiento de segmentos.**

## 6.5.6. Los cuatros segmentos de la memoria

Uno puede pensar que el tamaño de los segmentos sea insuficiente, bien porque el programa es muy grande, hay muchos datos, o ambas cosas a la vez. Esto no constituye ningún problema, pues el contenido de los registros de segmento puede ser alterado dinámicamente durante la ejecución del programa, haciendo uso si es necesario de toda la memoria.

El microprocesador 8086 crea una dirección física de 20 *bits* combinando un valor base de 16 *bits* desplazado, con un valor *offset* de 16 *bits*, almacenado en uno de los registros de segmentos. Cada uno de los registros segmento de 16 *bits*, el registro segmento de código (*CS*), el registro segmento de pila (*SS*), el registro segmento de datos (*DS*) y el registro segmento de datos extra (*ES*), contienen un valor que puede ser combinado con la dirección de desplazamiento de 16 *bits* especificada por el operando de la instrucción para formar la dirección de 20 *bits*.

La memoria por tanto, queda dividida en cuatro segmentos:

- **Segmento código:** Donde se almacenan las instrucciones.
- **Segmento pila:** Donde está localizada la pila.
- **Segmento datos:** Donde se encuentran los datos con los que se van a operar.
- **Segmento extra:** Un área de memoria extra, donde se pueden almacenar cadenas o datos.

El registro segmento código que se utilice para formar la dirección varía de acuerdo al tipo de instrucción que se esté ejecutando en la Unidad de Ejecución. La Figura 6.15 muestra una posible distribución en memoria de los distintos segmentos, por supuesto no siempre distribuidos así y en ese orden.

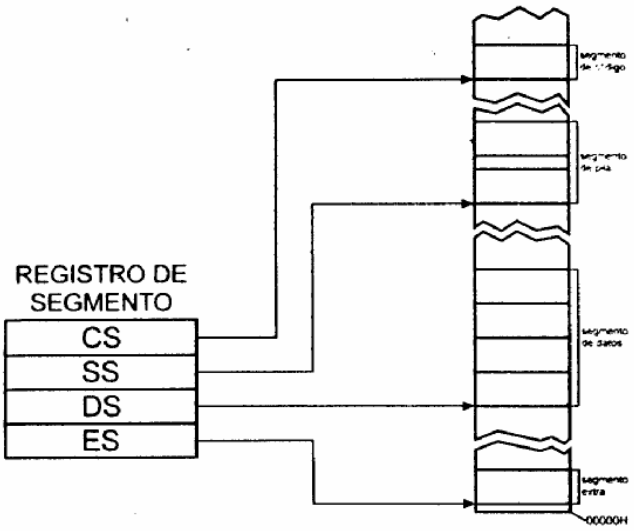


**Figura 6.15.** Distribución de los 4 segmentos.

## 6.6. Familia los microprocesadores de los ordenadores

Aunque hemos analizado el microprocesador de 16 *bits* 8086. actualmente existen varios modelos de microprocesadores de las marcas *Intel*. *AMD* y *Cyrix* desarrollados para los ordenadores personales. Estos microprocesadores ejecutan las instrucciones contenidas en la memoria al ritmo de una señal de reloj. Cuanto mayor sea su frecuencia, se obtendrá una mayor velocidad de trabajo. Según el tipo de instrucciones, el microprocesador puede operar de acuerdo a dos arquitecturas:

- **CISC** (***Complex Instruction Set Core***): Se traduce como 'juego completo de instrucciones' y su filosofía se basa en implementar un gran número de instrucciones en el microprocesador.
- **RISC** (***Reduced Instruction Set Core***): Se traduce como 'juego reducido de instrucciones' y su filosofía se basa en utilizar el menor número de instrucciones posibles. Aportando un fácil diseño y capacidad de realizar operaciones a gran velocidad a costa de utilizar programas de mayor extensión.

### 6.6.1. Microprocesadores de Intel

Los ordenadores compatibles *PC* con microprocesadores de *Intel* han utilizado tradicionalmente la arquitectura *CISC*, pero actualmente se están desarrollando soluciones mixtas. Para mantener la compatibilidad *PC* y beneficiarse de la arquitectura *RISC*. los microprocesadores dividen las instrucciones complejas en grupos reducidos para trabajar a nivel interno. En la Tabla 6.8 se muestran todos los microprocesadores desarrollados por *Intel* hasta la fecha para los ordenadores *PC* compatibles.

Estos microprocesadores difieren unos de otros en algunas características:

| Microprocesador | Bits interno | *Bits* externo | | Memoria (máx.) | Modos de operación | Caché interna | Paginación |
|---|---|---|---|---|---|---|---|
| | | Bus dir. | Bus Datos | | | | |
| 8086 | 16 | 20 | 16 | 1 Mb | Real | NO | NO |
| 8088 ✔ | 16 | 20 | 8 | 1 Mb | Real | NO | NO |
| 80186 | 16 | 20 | 16 | 1 Mb | Real | NO | NO |
| 80188 ✔ | 16 | 20 | 8 | 1 Mb | Real | NO | NO |
| 80286 | 16 | 24 | 16 | 16 Mb | Real, Protegido | NO | NO |
| 80386SX | 32 | 32 | 16 | 4 Gb | Real, Protegido, V86 | NO | SI |
| 80386DX | 32 | 32 | 32 | 4 Gb | Real, Protegido, V86 | NO | SI |
| i486SX | 32 | 32 | 32 | 4 Gb | Real, Protegido, V86 | SI | SI |
| i486DX | 32 | 32 | 32 | 4 Gb | Real, Protegido, V86 | SI | SI |
| i486DX2 | 32 | 32 | 32 | 4 Gb | Real, Protegido, V86 | SI | SI |
| i486DX4 | 32 | 32 | 32 | 4 Gb | Real, Protegido, V86 | SI | SI |
| Pentium (P5), Pentium Pro (P6), Pentium MMX y Pentium II | 64 | 32 | 64 | 4 Gb | Real, Protegido, V86 | SI | SI |

✔ *Usado en aplicaciones específicas.*