MODELO DEL PENTIUM PARA EL PROGRAMADOR DE APLICACIONES

7.1. PROGRAMADOR DE SISTEMAS Y PROGRAMADOR DE APLICACIONES

En los procesadores avanzados es muy importante saber apreciar la diferencia existente entre los dos tipos de programadores habituales que participan en la construcción del software.

7.1.1. Programador de aplicaciones

Se encarga de crear el sistema lógico que soportan las aplicaciones del usuario. Ve la CPU como un conjunto de registros de trabajo que le permiten confeccionar dichas aplicaciones del usuario. Con estos registros puede manipular instrucciones, datos y direcciones además de otros elementos que le permitirán desarrollar sus programas.

Para este programador resultan transparentes los recursos de la CPU empleados para llevar a cabo la tarea de aplicación junto con otras distintas de acuerdo con un mecanismo de protección que controla los accesos entre las tareas entre sí, entro los objetos de la tarea y entre los accesos de las tareas y el sistema operativo incluso en la ejecución de determinadas instrucciones.

Los conocimientos que el programador de aplicaciones debe tener sobre la máquina son los imprescindibles para obtener el máximo rendimiento de las instrucciones usadas para resolver las aplicaciones. En el caso de emplear el lenguaje máquina, deberá conocer los registros internos accesibles para la manipulación de datos y direcciones, el repertorio básico de instrucciones y los modos de direccionamiento. También deberá manejar el modelo de programación del coprocesador matemático auxiliar.

7.1.2. Programador de sistemas

Es necesario que conozca profundamente la arquitectura detallada de la CPU para así optimizar todos los recursos, obteniendo en su funcionamiento la máxima potencia, seguridad y rendimiento. Debe conocer también las prestaciones de la memoria virtual, las características de protección del entorno, los mecanismos que posibilitan la conmutación de tareas, el tratamiento de interrupciones y excepciones, etc.

La misión de este programador es construir un sistema de explotación óptimo que sea capaz de soportar todas las aplicaciones previstas. Entre sus funciones más destacadas están:

- Organizar el sistema para el correcto tratamiento de las tareas pertenecientes a los diferentes usuarios.
- Confección de objetos para sistemas operativos, depuradores, compiladores ...
- Asignar a cada tarea su nivel de privilegio y un sistema de protección adecuado.
- Organizar toda la memoria y el procesador para que de este forma las tareas consigan un mejor rendimiento.

Los microprocesadores Pentium disponen de una serie de registros y recursos especiales, denominados "del sistema", que se encargan de gestionar el funcionamiento general, aprovechando todas las prestaciones.

La complejidad de estos microprocesadores ha hecho necesaria la construcción de herramientas para el desarrollo de software. Estas herramientas varían dependiendo del tipo de progre

mador. Así, Intel inicialmente diseñó una herramienta lógica, denominada BINDER, encargada de la generación de aplicaciones, sencilla de manejo, pero que no permite el acceso a los mecanismos del sistema, haciéndola idónea para el programador de aplicaciones. El BUILDER por el contrario, es otra poderosa herramienta destinada a la construcción de sistemas, que posibilita el control de todos los recursos de la CPU, por lo que requiere ser usada por los programadores de sistemas, capaces de describir con detalle la estructura global que se precisa implantar.

7.2. REGISTROS INTERNOS PARA EL PROGRAMADOR DE APLICACIONES

El Pentium dispone de 32 registros en su arquitectura interna de los cuales la mitad, es decir 16, son para uso del programador de aplicaciones, tal y como se pueden apreciar en la figura 7.1

Estos últimos se clasifican en cuatro grandes grupos:

- Registros de propósito general
- Registro Puntero de Instrucciones (EIP)
- Registro de Estado (o de Señalizadores)
- Registros de Segmento

Antes de pasar a explicar estos cuatro grupos, recordaremos muy brevemente las tres formas de trabajo existentes en el Pentium.

- Modo real: trabaja como el 8086 es decir, se usa la segmentación y al iniciar el sistema funciona de esta manera (16 bits).
- Modo protegido: este es el modo nativo con toda su potencia (32 bits). Puede trabajar con memoria real, memoria virtual, sistemas de protección, multitarea, paginación ...
- Modo de manejo del sistema.

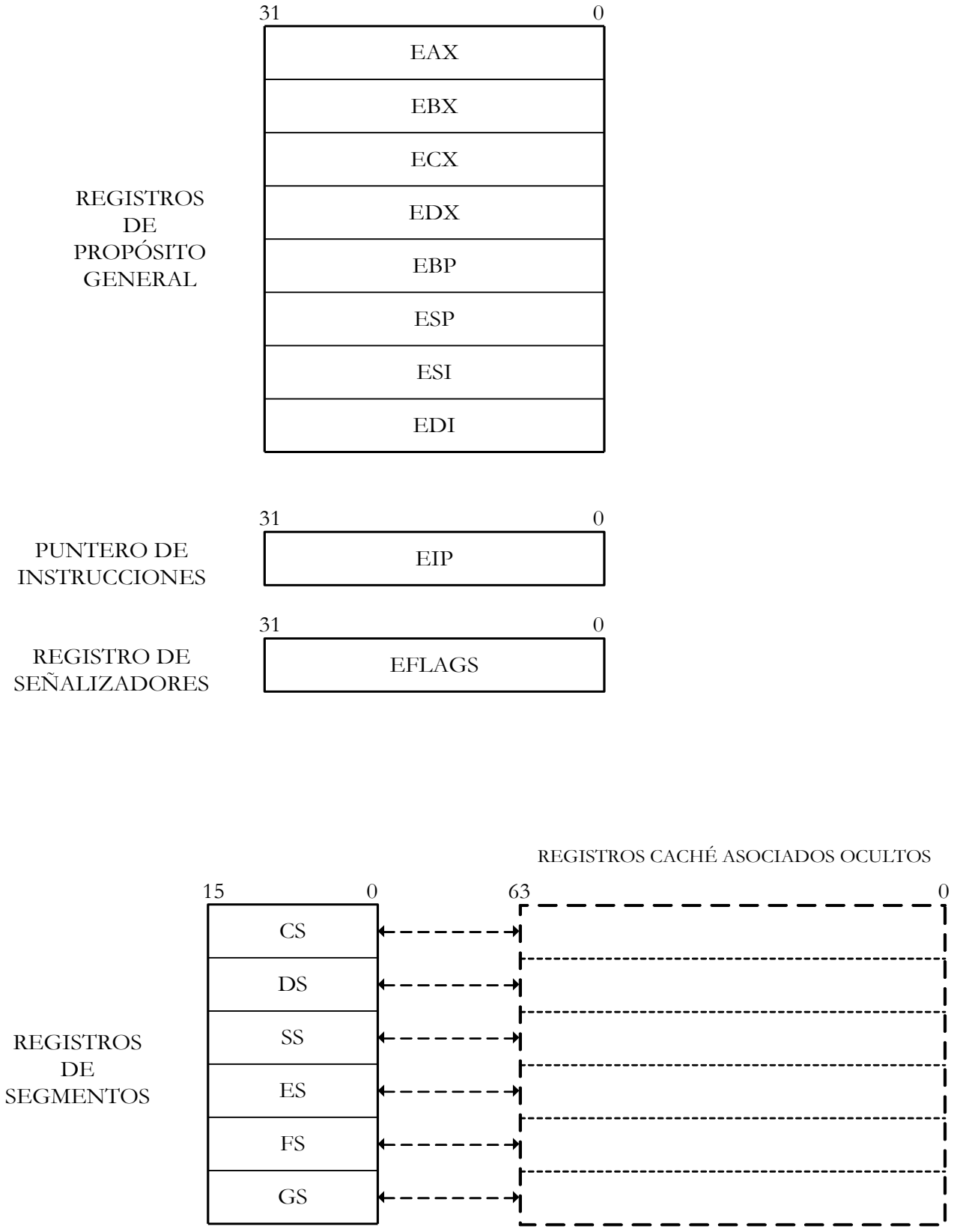

Figura 7.1 – Esquema general de los 16 registros disponibles para el programador de aplicaciones.

7.2.1. Registros de propósito general

Son los mismos registros que tenía el 8086 de 16 bits pero ampliados a 32 bits. Este grupo consta de ocho registros capaces de trabajar con información de 32 bits cuando utilizan todo su tamaño aunque también pueden manejar 16 bits e incluso, 4 de estos registros pueden manejar 8 bits.

Cuando se hace referencia a uno de los registros que trabaja con 32 bits, se pone la letra "E" de extendido precediendo al nombre habitual del registro de 16 bits de los microprocesadores 8086 y 80286. Los nombres de cada uno de estos ocho registros son:

- **EAX**: Acumulador
- **EBX**: Base
- **ECX**: Contador
- **EDX**: Datos
- **ESP**: Puntero de pila
- **EBP**: Puntero de base (Base Pointer)
- **ESI**: Índice fuente (Snack Pointer)
- **EDI**: Índice destino

Cuando se accede únicamente a los 16 bits de menos peso de estos registros, se designan por AX, BX, CX, DX, SP, BP, SI, DI, respectivamente.

También son accesibles, de forma independiente, los dos bytes de menos peso de los registros AX, BX, CX y DX. Dentro de estos 16 bits de menos peso, si accedemos al byte de menos peso se le asigna con AL, BL, CL y DL, respectivamente, mientras que si accedemos al de más peso se le referencia con AH, BH, CH y DH, tal y como se representa en la figura 7.2

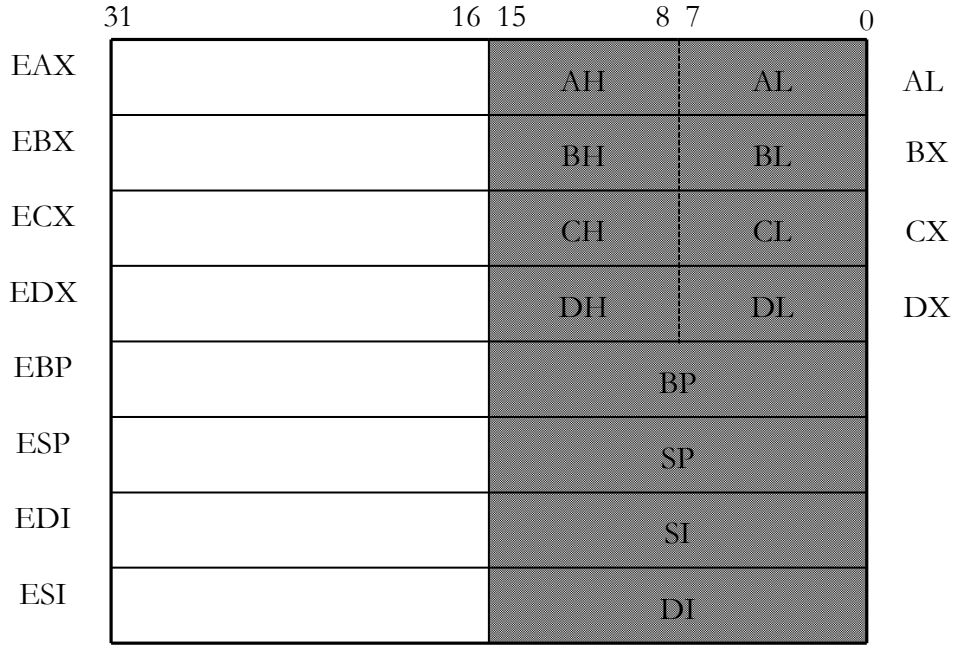

Figura 7.2 – Formatos que pueden adoptar los registros que corresponden al grupo destinado a propósitos generales.

Los registros de propósito general pueden usarse tanto para almacenar datos como instrucciones. En este último caso, el contenido del registro es un desplazamiento que apunta a una dirección de memoria.

Teniendo en cuenta todos los posibles formatos de los registros de este grupo, quedan disponibles ocho registros de 32 bits, otros ocho de 16 bits y otros ocho de tamaño byte. Por tanto, cuando se emplean para contener datos, posibilita al Pentium operar indistintamente sobre bytes, palabras, dobles palabras y cuádruples palabras.

Los registros EAX, EBX, ECX y EDX, se emplean fundamentalmente en operaciones generales como las lógicas, las aritméticas... Es importante mencionar en este punto, que Intel ha procurado mejorar la simetría de los registros en los microprocesadores avanzados, es decir, favorecer la flexibilidad de usar cualquiera de ellos en la mayoría de las instrucciones. En microprocesadores anteriores, bastantes instrucciones requerían el uso específico de ciertos registros para contener operandos o el resultado.

Por lo general el registro EAX, muy utilizado en los procesadores Intel, se emplea como Acumulador en instrucciones lógico-aritméticas y las relacionadas con las transferencias entre la memoria y la CPU.

EJEMPLO 1

MOV AH, AL

El byte de menos peso del Acumulador se mueve al byte AH, que es el de más peso

---

El registro EBX contiene la base de la dirección donde empieza una estructura datos, bien sea un array, una pila...

El registro ECX es habitual utilizarlo como contador en instrucciones que contengan iteraciones, es decir, que se repitan distintas veces, por ejemplo en la operación de notación ROR.

El registro EDX se utiliza como apoyo al EAX. Si por ejemplo el valor del producto no cabe en EAX se agranda con este registro. También para operaciones de E/S, posiciones de E/S que van periféricos o vienen de periféricos.

EJEMPLO 2

```
MOV ECX, 08h.
Start:
   IN AL, 70h.
   MOV [EBX], AL.
   INC EBX.
   LOOP Start.
```

Se carga ECX con 08h, se define el comienzo del bucle, se lee AL vía puerto 70h, se escribe el contenido de AL en el lugar de memoria [EBX], se incrementa en uno la dirección EBX y por último se llevan a cabo 8 repeticiones (hasta que ECX valga 0).

Los registros apuntadores ESP y EBP, sirven para controlar el direccionamiento de la pila y almacenar desplazamientos relativos a la pila en curso.

Las operaciones en la pila las soportan tres registros diferentes que son los que exponemos a continuación:

1. Registro de segmento de pila (SS). Especifica las características del segmento de pila que reside en memoria. El número de pilas en un sistema, está limitado sólo por el máximo número de segmentos. Por tanto, una pila puede tener hasta 4 GBytes de longitud que es el máximo tamaño de un segmento. El registro SS, del que hablaremos en el apartado 7.2.4., lo usa el procesador para todas las operaciones de la pila.

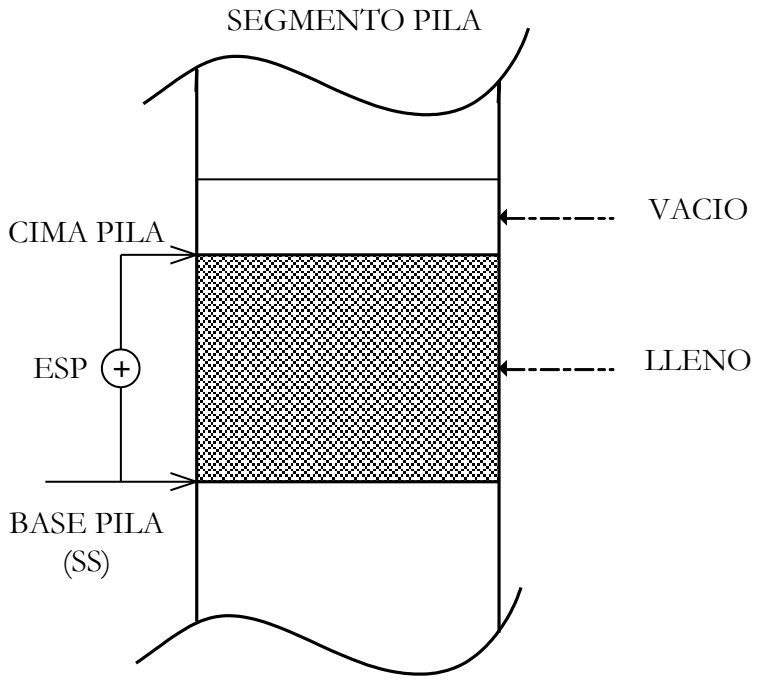

Figura 7.3 – Figura del funcionamiento de la pila SS

2. Registro puntero de pila (ESP). Contiene el desplazamiento de la cima de la pila en el segmento de la pila actual. Lo usan las operaciones PUSH y POP, las llamadas a subrutinas, el retorno, las excepciones y las interrupciones. Cuando se introduce un elemento a la pila, el procesador decrementa el puntero ESP, y escribe el elemento en la cima de la pila. Cuando se saca un elemento se hace la operación contraria, es decir, incrementar ESP.

3. Registro puntero base de la pila (EBP). Se usa normalmente par a acceder a estructuras de datos pasadas en la pila. Cuando el registro EBP se usa para direccional memoria, el segmento de pila en curso es referenciado. Este registro apunta a la base de la pila y cuando existen rutinas hace el papel de ESP para así no modificar el valor de este último.

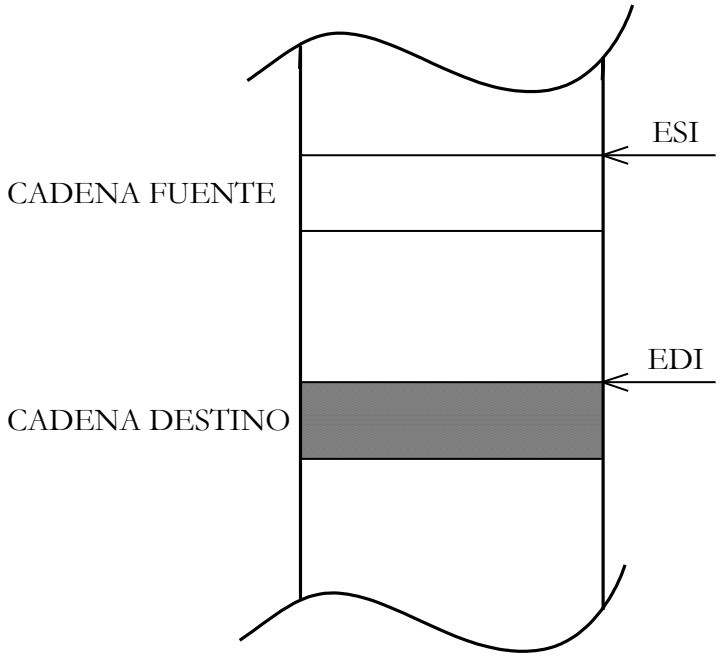

Figura 7.4 – Figura del funcionamiento de los registros ESI y EDI

Por último, los registros ESI y EDI, contienen valores índice usados en la exploración de grandes conjuntos de datos, cadenas, arrays, etc...y admiten la posibilidad de incremento y decremento automático de su valor para relaciones fuente y destino, como se puede apreciar en la figura 7.4.

7.2.2. EIP: Registro puntero de instrucciones

Es un registro de 32 bits que almacena el desplazamiento que hay que añadir a la base del segmento de código para obtener la dirección donde está la siguiente instrucción, es decir, la dirección que el procesador tiene que ejecutar a continuación. Por tanto, el registro EIP contiene el valor de dicho desplazamiento, mientras que la base se obtiene a partir del contenido del registro de segmento de código CS.

El EIP no está directamente disponible para el programador , lo gobierna implícitamente el control de transferencias de las instrucciones, las interrupciones y las excepciones.

Figura 7.5 – El Puntero de Instrucciones admite el tamaño de 32 bits (EIP) y el de 16 bits (IP), cuando soporta el direccionamiento "reducido".

Este registro puede trabajar en dos modos:

- En modo nativo, que se acaba de describir, el cual recibe el nombre de EIP y posee 32 bits.
- En modo real, cuando se emplea este tipo de direccionamiento reducido, compatible con los procesadores 8086 y 80286, sólo se precisan de 16 bits para especificar el desplazamiento, que son los dos bytes de menos peso de EIP, y que se denominan IP. (figura 7.5)

En la memoria segmentada la dirección de la instrucción en curso se halla sumando el desplazamiento a la base donde comienza el segmento del código, como podemos ver en la figura 7.6

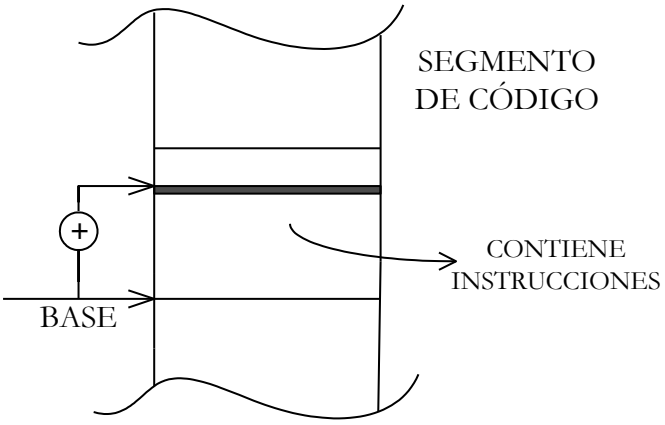

Figura 7.6 – Funcionamiento de la memoria segmentada

7.2.3. Registro de estado o señalizadores

Este registro también conocido como el registro EFLAGS consta de 32 bits de los cuales la mayoría son señalizadotes de estado, controlados por la ALU (acarreo, paridad, acarreo auxiliar, cero, signo y sobrepasamiento), actuando los restantes como señalizadotes del sistema, ligados al mecanismo de protección y a otros recursos de que dispone el sistema de explotación cuya misión será mejor interpretada a medida que se profundice en el estudio del procesador (figura 7.7)

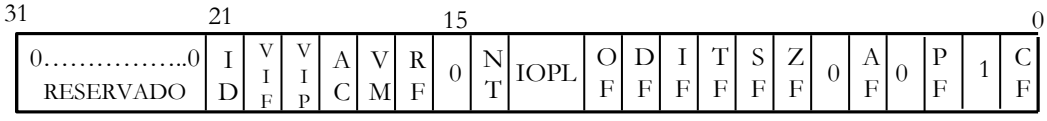

Figura 7.7 – Registro EFLAGS

A continuación, pasaremos a explicar cada uno de los bits que forman parte del registro de estado siguiendo el orden de menos peso a más peso:

- **CF**: Señalizador de acarreo en el bit más peso (en el más significativo). Si trabajamos en modo real es en el bit 16 y si trabajamos en modo protegido es en el bit 32.

    - 1: Si hay acarreo en las operaciones de suma aritmética.
    - 0: En las restas cuando hay llevada en el bit de más peso. También se utiliza este bit en operaciones de desplazamiento y rotación.

- **PF**: Bit de paridad impar.

    - 1: Para generar la paridad impar con los bits que conforman el resultado de una operación
    - 0: En caso de no producirse paridad impar.

- **AF**: Señalizador de acarreo auxiliar (o intermedio).

    - 1: Si ha habido acarreo en el bit 3 del resultado (contando sería el 4º bit). Se utiliza en las operaciones BCD.
    - 0: En caso de no haber acarreo auxiliar.

- **ZF:** Señalizador de cero.

    - 1: Si todos los bits del resultados son cero. Es muy útil al comparar dos registros, ya que si tienen el mismo contenido ZF pasa a valer 1
    - 0: En caso de que el resultado no sea 0

- **SF**: Señalizador de signo.

    - 1: Si el bit de más peso del resultado de la operación es 1.
    - 0: Si el bit de mas peso del resultado de la operación en 0.

- **TF**: Excepción al terminar la ejecución de la instrucción.

    - 1: Provoca una excepción al completarse la ejecución de la instrucción en curso. Posibilita la depuración de los programas al permitirse la ejecución paso a paso, es decir instrucción por instrucción.
    - 0: No hay excepción

    Si un programa de aplicación dispone de un flag TF utilizando una instrucción POPF, POPFD, o IRET, se genera una "limpieza de excepción" después de la instrucción que sigue a la de estas.

- **IF**: Flag de habilitación de interrupciones.

    - 1: Permite el reconocimiento de las peticiones de interrupción mascarables provocadas por la activación de la patita INTR del Pentium. Las no mascarables se atienden siempre ya que son de máxima prioridad.
    - 0: Prohíbe el reconocimiento de la interrupción externa y no la atiende.

    La señal IF no afecta a la generación de excepciones o interrupciones no mascarables como por ejemplo la interrupción NMI. El CPL, IOPL y el estado del flag VME del registro de control CR4 determina si el flag IF puede ser modificado por las instrucciones CLI, STI, POPF, POPFD e IRET.

- **DF**: Flag de dirección de exploración de las cadenas de caracteres o strings.

    - 1: Postdecremento automático de los registros ESI, EDI (registros de propósito general), que direccional la cadena.
    - 0: Postincremento automático en ESI, EDI.

- **OF**: Flag de sobrepasamiento (u overflow)

    - 1: En operaciones con números enteros con signo se activa si el resultado es muy grande (es positivo) o muy pequeño (si es negativo). Indica resultados erróneos por ejemplo si existe acarreo en el penúltimo bit.
    - 0: Si no existe sobrepasamiento

- **IOPL**: Nivel de privilegio de las entradas y salidas (input/output). Es un campo de 2 bits que se emplean en modo protegido y determina:

    1. El nivel de privilegio a partir del cual se pueden ejecutar las instrucciones de E/S sin generar error. Las instrucciones de E/S son las siguientes:
        - IN, entrada
        - OUT, salida
        - INS, string como entrada
        - OUTS, string como salida
        - CLI, pone a 0 el bit IF
        - STI, pone a 1 el bit IF

        Únicamente las instrucciones input y output pueden usarse para acceder a esta memoria.

    2. El nivel de privilegio de privilegio que permite la alteración del señalizador IF al cargar el registro EFLAGS.

    Los valores que se pueden tomar son:
    - 11: nivel 3 (pueden acceder todos)
    - 10: nivel 2
    - 01: nivel 1
    - 00: nivel 0 (únicamente puede acceder el S0)

    POPF e IRET pueden modificar el campo IOPL solamente cuando se ejecutan con el máximo nivel de privilegio, o sea, el 0. Una conmutación de tarea puede alterar siempre el IOPL al cargarse el nuevo EFLAGS desde el segmento de estado de la nueva tarea como se explicará posteriormente.

    El IOPL es igualmente uno de los mecanismos que controla la modificación del flag IF y el manejo de interrupción en la función virtual 8086 cuando las extensiones de la función virtual están vigentes (el flag VME en registro de control CR4 está dispuesto).

- **NT**: Tarea anidada. Este señalizador se activa y desactiva automáticamente al producirse una conmutación de una tarea.

    - 1: La tarea en curso está anidada con la anterior, es decir, luego hay que volver a ésta.
    - 0: La conmutación de tareas es libre

El flag puede ser explícitamente activado o desactivado mediante las instrucciones POPF/POPFD. Sin embargo, la alteración del estado de este flag puede originar inesperadas excepciones en programas de aplicación.

- **RF**: Flag de reanudación. Su activación provoca la ejecución de la siguiente instrucción cuando se produce un fallo de depuración en una instrucción, es decir, se ignora el fallo producido en la depuración.

    - 1: Se ignoran los puntos de parada
    - 0: No se ignoran los puntos de parada

- **VM**: Modo virtual – 86. Trabajando en modo protegido, se permite que algunas tareas trabajen en Modo Real.

    - 1: Estando el procesador en Modo Protegido, se pasa a Modo Virtual 86
    - 0: No se hace nada

    Este bit puede activar por medio de la instrucción IRET, cuando se está en Modo Protegido si el nivel de privilegio en curso CPL = 0. También por medio de una conmutación de tarea en cualquier nivel de privilegio.

    El señalizador VM no queda afectado por la instrucción POPF. Sin embargo, cuando se ejecuta PUSHF, se pone a 0 este bit, incluso cuando se está trabajando en Modo Virtual-86.

- **AC**: Bit de chequeo de alineamiento.

    - 1: Se produce una excepción para alinear una palabra
    - 0: No hay excepción

    Las direcciones de palabra de palabras de 2 bytes deben ser múltiplos de 2, las de 4 bytes deben serlo de 4 y las de 8 bytes, múltiplos de 8. Si un programa de aplicación con el mínimo nivel de privilegio, es decir, nivel de privilegio 3, tiene una palabra desalineada y este flag AC = 1, se produce una excepción en concreto la excepción número 17. Las referencias de memoria que por defecto tienen un nivel de privilegio 0 no generan esta excepción, incluso cuando son ejecutadas en modo usuario.

    Si el programador no pone los bits de datos alineados, el bus trabajará más, es decir, necesitará más ciclos para leer lo mismo que si estuviesen alineados. Como hemos dicho anteriormente, si AC = 1 y se produce un fallo de alineamiento, se genera una excepción. Por el contrario, si AC = 0, el acceso se realiza en más ciclos, es decir, no se produce excepción alguna.

    Al ponerse a 1 los flags AC y AM en el registro de control CR0 se habilita el chequeo de alineamiento de las referencias de memoria.

    La excepción por chequeo de alineamiento puede ser empleada para chequear el alineamiento de datos. Esto es útil cuando se intercambian datos con otros procesadores, los cuales requieren todos los datos para ser alineados.

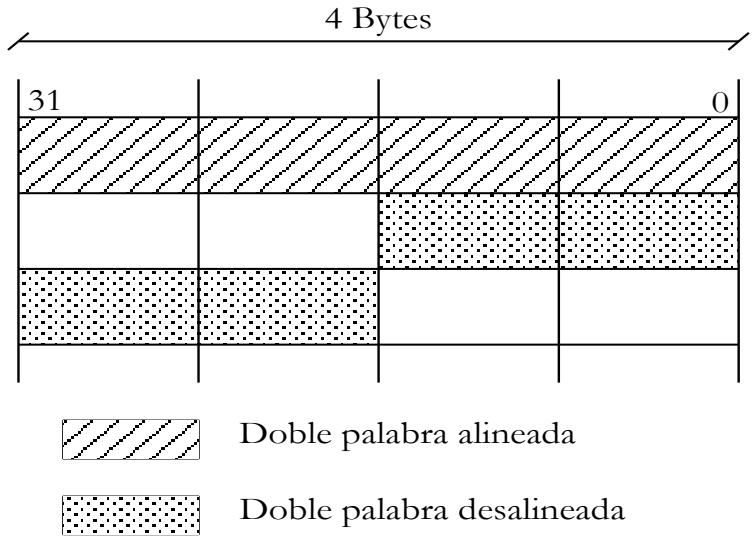

Figura 7.8 – Alineamiento de la memoria

En la figura 7.8, podemos observar una doble palabra alineada y una doble palabra desalineada. En caso de que la doble palabra estuviera alineada, se cogen los 4 Bytes y se colocan en el bus de datos consiguiendo así una lectura rápida. En el caso opuesto, es decir, si la doble palabra se encuentra desalineada, habrá que leer en dos accesos por lo que se produce una pérdida considerable en el rendimiento.

Por último, citar que Intel permite que existan las palabras desalineadas pero Motorola lo prohíbe.

- **VIP**: Interrupción (mascarable) virtual pendiente. Trabaja en combinación con el flag VIF para que cada tarea en modo virtual disponga se su flag IF. Así se aceleran notablemente las interrupciones y las instrucciones CLI y STI no provocan excepción. Se activa por software para indicar que una interrupción está pendiente. El procesador lee este flag pero nunca lo modifica. El procesador sólo reconoce el flag VIP cuando el flag VME o el PVI en el registro de control CR4 están activados y el IOPL es menor que 3.

    - 1: Interrupción pendiente
    - 0: No hay interrupción pendiente

- **VIF**: Interrupción virtual. Es un flag equivalente a IF cuando se trabaja en modo virtual 8086. Este flag se utiliza con el flag VIP. El procesador sólo reconoce el flan VIF cuando bien el flag VME o el PVI en el registro de control CR4 están activados y el IOPL es menor que 3.

- **ID**: Bit de identificación. El flag ID indica si el Pentium soporta la instrucción CPUID que sirve para su identificación. Mediante CPUID se informa sobre las características más importantes del procesador. Le informa al software acerca del modelo de microprocesador en que se está ejecutando. Un valor cargado en EAX antes de ejecutar esta instrucción deberá retornar CPUID. Si EAX = 0, se cargará en dicho registro el máximo valor de EAX que se podrá utilizar en CPUID (para el Pentium este valor es 1). Además, en la salida aparece la cadena de identificación del fabricante contenido en EBX, ECX y EDX. EBX contiene los primeros cuatro caracteres , EDX los siguientes

cuatro, y ECX los últimos cuatro. Para los procesadores Intel la cadena es "GenuineIntel".

- 1: El Pentium soporta la instrucción CPUID que sirve para la identificación de la CPU (número de serie, frecuencia con la que se trabaja ...)
- En caso contrario.

7.2.4. Registro de segmento

Intel ha incorporado en la arquitectura IA-32 la segmentación como sistema principal en la organización de la memoria. La segmentación favorece la programación estructurada y la modularidad, siendo una técnica apropiada para permitir la reubicación y soportar una estructura de protección muy segura y flexible.

Los segmentos son zonas de la memoria de tamaño variable que contienen el mismo tipo de información. Así, hay tres tipos de segmentos principalmente:

- De pila
- De código
- De datos

El Pentium controla en cada instante 6 segmentos a los que direcciona a través de los Registros de Segmento (RS). Esto no significa que sólo pueda manejar 6 segmentos, sino que, si desea acceder a otro que no está referenciado por dichos registros, deberá cargarse previamente, en uno de ellos, el valor correspondiente al nuevo segmento.

Para direccionar la ubicación de los segmentos, el Pentium dispone de 6 registros de 16 bits que se muestra a continuación en la figura 7.9.

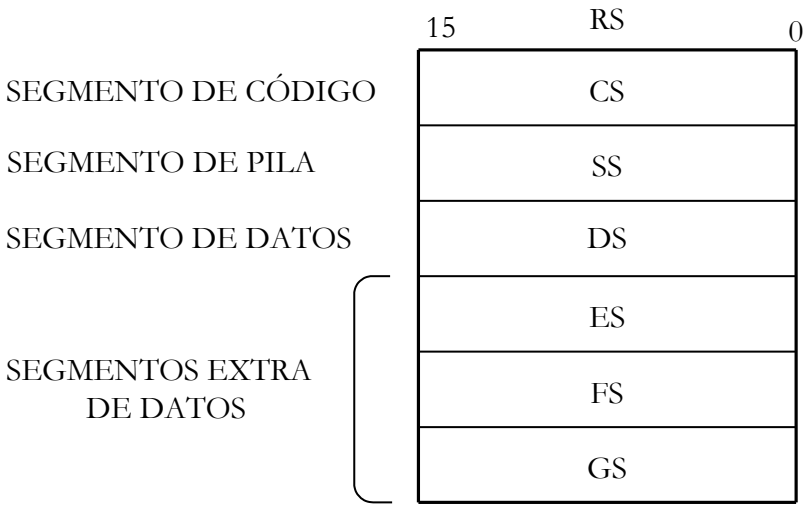

Figura 7.9– Registro de segmento en el Pentium

Desde el punto de vista del programador de aplicaciones, los 6 registros de segmento materializan, en cada momento, los segmentos que es capaz de identificar y manipular la CPU.

La dirección lógica o virtual de todo elemento accesible en la memoria, está formada por un puntero que consta de los siguientes campos:

- **Selector**: Es un valor de 14 bits, contenido en cada uno de los 6 registros de segmento que identifica al segmento y en concreto la dirección de su base. Los dos bits de menos peso de los registros de segmento no intervienen en el selector ya que contienen el campo RPL que indica el nivel de privilegio del peticionario, como se puede apreciar en la figura 7.10. El bit de menos peso de los 14 bits del campo selector es el TI (Índice de Tabla), que indica que la Tabla de Descriptor es local (TI=1) o global (TI=0), ayudando a localizar el segmento determinado en la memoria.

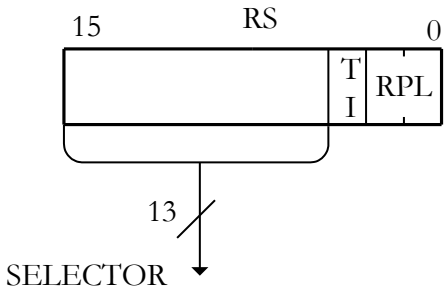

Figura 7.10 – Contenido del registro de segmento

- **Desplazamiento**: Es un valor que se añade a la base del segmento para localizar la dirección que hay que acceder en él. El tamaño de desplazamiento determina su longitud máxima que, en el caso del Pentium, es de 2^{32} = 4 GBytes, en el modo protegido, mientras que en el modo real son 2^{16} = 64 KBytes.

Cada vez que el procesador ejecuta un programa, utiliza estos dos campos. En concreto emplea el campo selector para calcular la base del segmento, es decir , la dirección donde comienza y el campo desplazamiento es el valor que suma a la base para determinar la posición a acceder tal y como se muestra en la figura 7.11.

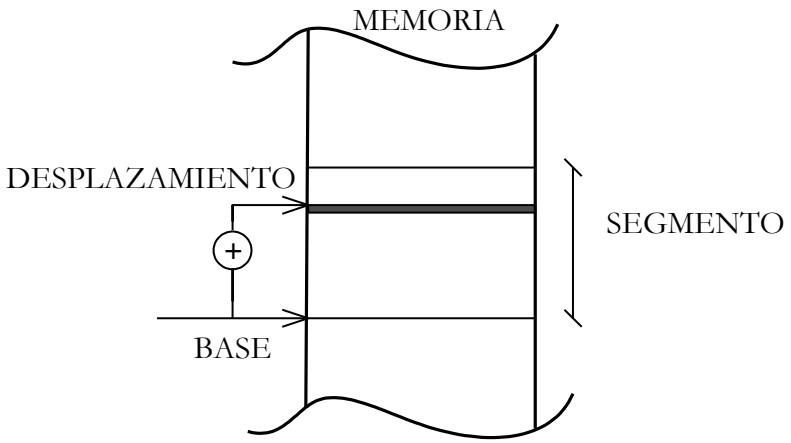

Figura 7.11 – Dirección lógica de un elemento (desplazamiento)

A continuación, después de ver el funcionamiento general para todos ellos, explicamos cada uno por separado:

- **CS**: El registro CS (Segmento de Código), contiene en cada momento el segmento de código que está ejecutando la CPU, es decir el segmento en curso. El desplazamiento que hay que añadir a la base del CS reside el Registro Puntero de Instrucciones EIP, explicado detalladamente en el apartado 7.2.2.

- **SS**: El registro SS (Segmento de Pila), guarda el valor del selector del segmento de pila en curso. El registro ESP contiene el desplazamiento que debe añadirse a la base del SS para determinar la cima, donde se cargan y descargan los datos.

- **DS**: El registro DS (Segmentos de Datos), soporta el valor del selector del DS y el desplazamiento viene especificado en el modo de direccionamiento usado en la instrucción para expresar operandos y el resultado.

EJEMPLO 3

MOV EAX, (5555).

Se carga EAX con el contenido de las posiciones de memoria del segmento DS que comienza en la dirección formada por su base más el desplazamiento 5555. Cuando no se explicita en el segmento de Datos se sobreentiende que es DS. Si fuese el segmento de Datos GS, la instrucción sería MOV EAX, GS : 5555.

Sobre este último punto, mencionar que el Pentium dispone de otros tres segmentos de datos activos, además del referenciado por DS, que reciben el nombre de segmentos "extra" y se denominan ES, FS y GS. De esta forma, se puede acceder a cuatro segmentos de datos sin alterar el valor de los registros de segmento, lo que redunda considerablemente en la velocidad de procesamiento de la CPU, como se analizará posteriormente.

En la figura 7.12, se muestra el mapa de memoria en el que se han especificado los seis segmentos activos para la CPU, en un momento determinado.

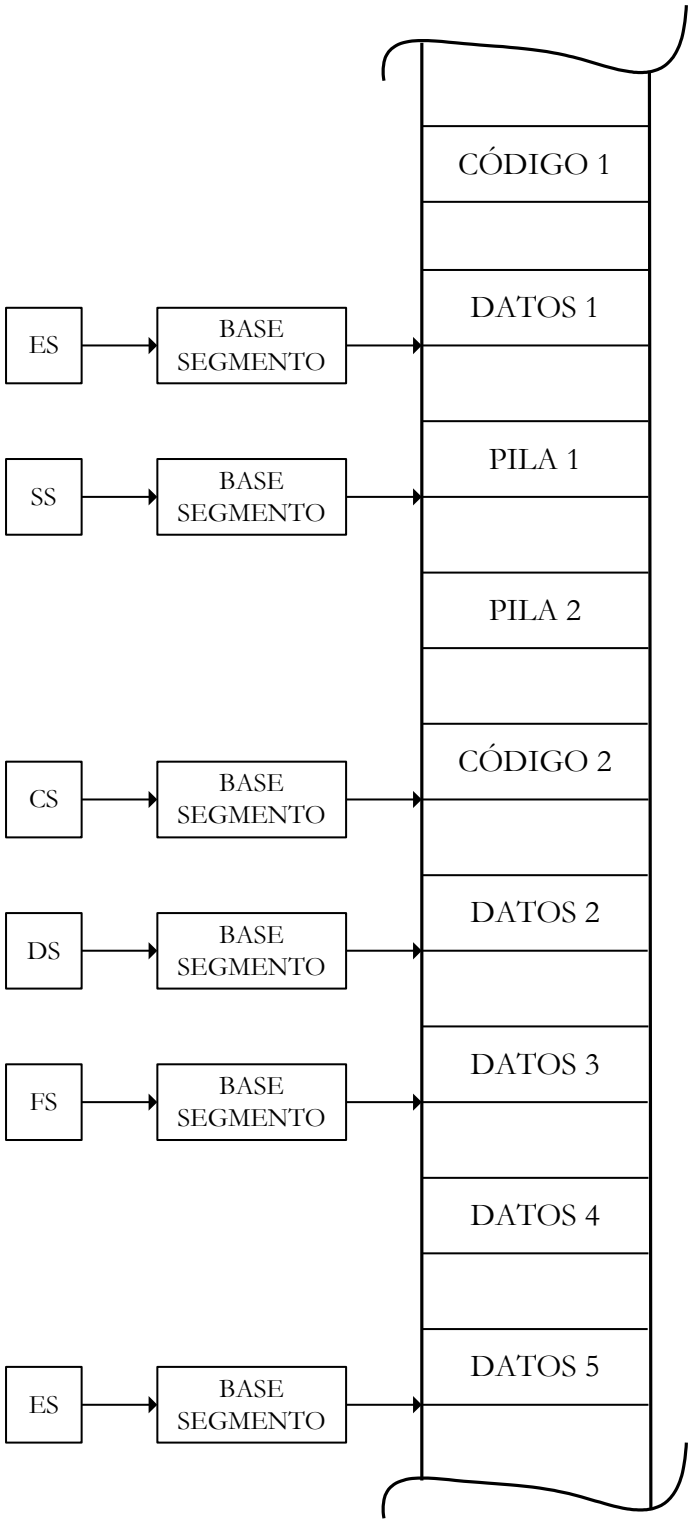

Figura 7.12 – A través de los registros de segmento, la CPU tiene activos a seis segmentos en cada instante. Para utilizar otros segmentos hay que modificar los selectores correspondientes contenidos en los registros de segmento.

7.3. SEGMENTACIÓN EN MODO REAL

Cuando el Pentium funciona en modo real (monotarea), compatible con el 8086, y sin tipo alguno de protección ni posibilidad de manejo de memoria virtual, un segmento queda definido básicamente por los siguientes elementos:

1. **Base** o dirección de comienzo de 20 bits.

2. **Desplazamiento** o tamaño de 16 bits. El tamaño máximo que se admite en este modo para mantener la compatibilidad con el 8086, es de 64 KBytes y la capacidad máxima de la memoria principal sólo es de 1 MByte.

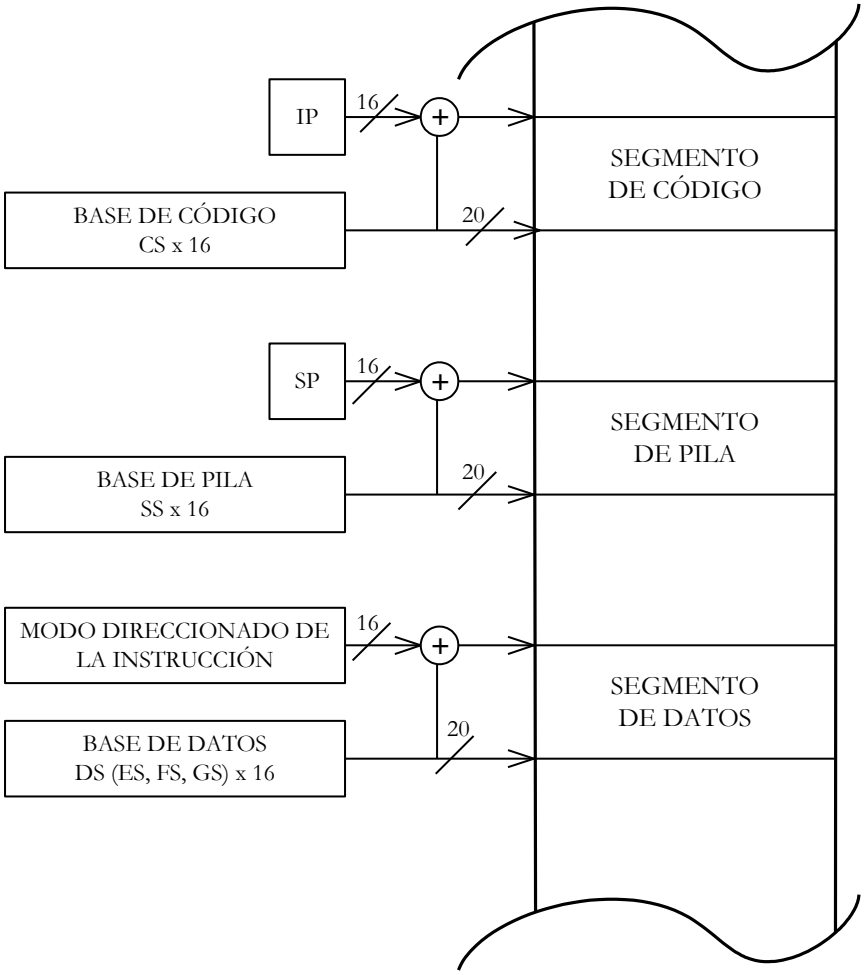

Figura 7.13 – Segmentación en modo real. En este modo se accede a elementos de memoria multiplicando por 16 el valor del registro de segmento y añadiendo un desplazamiento de 16 bits al resultado.

En modo real todo segmento de código, datos o pila está especificado por una dirección lógica, compuesta por dos campos de 16 bits cada uno.

- Selector: Referencia la base del segmento, la cual se deduce a partir del valor contenido en el registro de segmento apropiado. Como el 8086 sólo maneja una memoria de 1 MByte (2^{20}) de capacidad máxima, para obtener la base del segmento se añaden cuatro ceros a los 16 bits del registro de segmento o selector, es decir, se multiplica dicho valor binario por 16.

- Desplazamiento: El tamaño máximo del segmento en el 8086 es de 64 KBytes , por lo tanto bastan 16 bits para expresar el desplazamiento que hay que añadir a la base. En el caso del segmento de código, el desplazamiento lo almacena IP, que está formado por los 16 bits de menos peso de EIP. El desplazamiento correspondiente al segmento de pila está guardado en SP, y el del segmento de datos lo expresa el modo de direccionamiento de los operandos o del resultado en la instrucción en curso, por ejemplo (MOV AX, ES : 555 hex).

Por tanto una dirección quedaría: Dirección efectiva = RS x 16 + Desplazamiento. Esto queda especificado en la siguiente tabla. Dependiendo de si es un segmento de código, de pila o de datos, se conseguirá la base utilizando CS, SS, DS... tal y como se muestra a continuación.

	BASE	**DESPLAZAMIENTO**
CÓDIGO	CS X 16	IP
PILA	SS X 16	SP
DATOS	DS X 16 ES X 16 FS X 16 GS X 16	DESPLAZAMIENTO

Tabla 7.1. – Tabla para el modo real

7.4. SEGMENTACION EN MODO PROTEGIDO

Cuando el Pentium trabaja en modo protegido (multitarea), un segmento queda caracterizado por tres parámetros fundamentales que son comprobados automáticamente por el sistema de protección cada vez que se utiliza. Dichos parámetros son:

1. **Base**, es la dirección lineal donde comienza el segmento. Esta formada por 32 bits, que es la longitud de la dirección de la memoria física que puede alcanzar un tamaño máximo de 2^{32} = 4 GBytes.

2. **Límite**, consta de 20 bits que determinan con exactitud el tamaño del segmento usado por el programador y en el que residen informaciones válidas. Si está expresado en bytes, el lí

mite máximo sería de 2^{20} = 1 MByte y si está expresado en páginas de 4 KByte, un segmento puede ser tan grande como la memoria principal, es decir 4 GBytes.

3. **Atributos** o derechos de acceso, se trata de un campo de 12 bits, que proporciona las características relevantes del segmento como:

    - Tipo de segmento, admitiendo las variantes de legible, escribible, ejecutable o una combinación de estos.

    - Nivel de privilegio, que oscila entre 0 y 3. Es el grado de seguridad que tiene el contenido del segmento en el sistema.

    - Indicadores sobre aspectos relacionados con la gestión de la memoria virtual, como el que indica si el segmento se halla cargado o no en la memoria física.

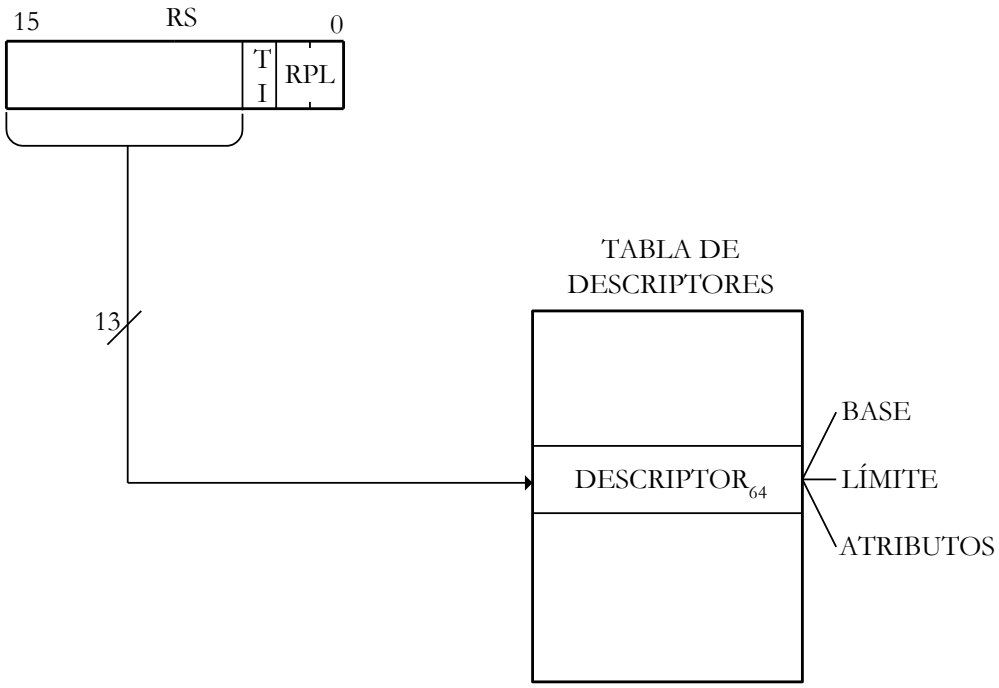

Figura 7.14 – Se accede a la tabla de descriptores consultado el selector y comprobando el índice de tabla.

Haciendo mención a lo comentado en la figura anterior, la figura 7.14, si **TI** (Indice de Tabla) es 0, se accederá a la tabla de descriptores global (GDT); en caso contraio, si es 1: se accederá a la tabla de descriptores local (LDT).

Como el índice es de 13 bits el número máximo de descriptores en la tabla de descriptores es de 2^{13} = 8 KByte. Por lo tanto se puede apuntar a 16 segmentos distintos. El máximo de memoria virtual que puede utilizar el Pentium sería 16 K descripores X 4 GBytes = 64 TBytes.

Al conjunto de los parámetros base (32 bits), límite (20 bits) y atributos (12 bits) que se utilizan para definir un segmento se denomina **descriptor de segmento** y tiene una longitud de 64 bits.

En modo protegido para obtener el descriptor de segmento, el Pentium utiliza el valor de registro de segmento, para acceder a unas tablas residentes en memoria principal. Como necesita disponer directamente de estas características, a cada segmento se le asocia un registro caché ultrarrápido (que se les denomina "ocultos" o "invisibles") que no son accesibles ni en lectura ni en escritura por el programador de sistemas, ni por el de aplicaciones. Es la CPU quien se encarga de gestionarlos automáticamente cada vez que se modifica el contenido de un registro o segmento.

Cuando se carga un registro segmento, la CPU busca automáticamente en las tabla de descriptores residentes en memoria principal, la base, el limite y los atributos del segmento referenciado y los carga en el registro caché asociado. A partir de aquí, todo acceso a dicho segmento, se realizará utilizando los datos que se encuentran en el registro caché, de acceso muy rápido. Cuando se modifica el valor de algún registro de segmento, surge una penalización en el tiempo ya que se tiene que actualizar la caché. Por eso hay cuatro registros de datos (ES, DS, FS, y GS) ya que lo normal es cambiar los segmentos de datos y no los de código. Todo ello se muestra gráficamente en la figura 7.15.

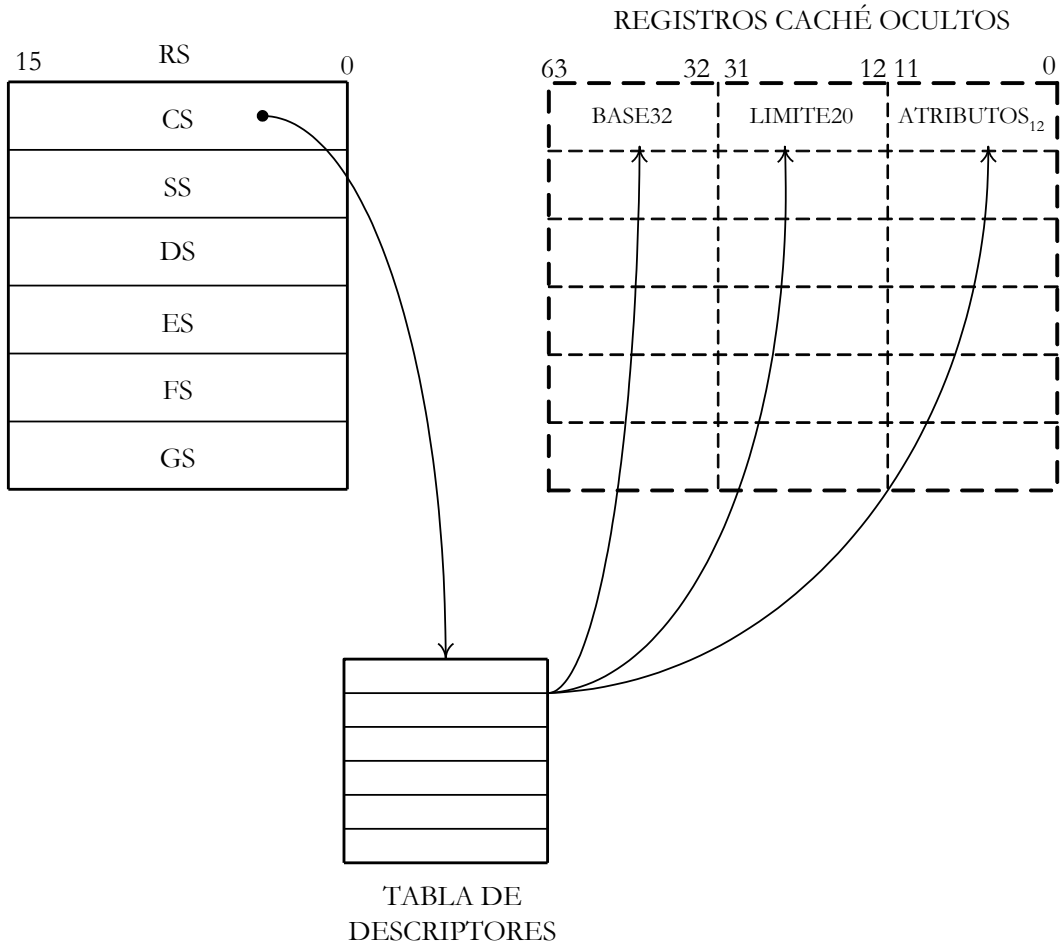

Figura 7.15 – En modo protegido, los registros de segmento actuan como selectores "visibles", con los que se accede a tablas en las que se hallan los parámetros que definen al segmento, es decir, la base, el límite y los atributos. Estos parámetros los carga automáticamente la CPU en los registros caché asociados, que son invisibles, es decir, inaccesibles al programador de sistemas y al de aplicaciones.

Como la carga de los registros caché invisibles consume un tiempo extra, se comprende que la instrucción PUSH DS se ejecute más rápidamente que POP DS, ya que la primera modifica el valor de DS y la CPU automáticamente busca en las tablas los valores que definen el nuevo segmento y los carga en el registro caché asociado a DS.

En modo real, también se usan los registros caché invisibles, siendo el valor de la base el del registro de segmento con cuatro ceros añadidos, el límite siempre es el máximo, o sea, 64 KBytes y todos los derechos de acceso están permitidos.

En capítulos posteriores, se estudiará en profundidad el mecanismo de direccionamiento en modo protegido . En este capítulo sólo se ha presentado la estructura de los registros disponibles por el programador de aplicaciones y su función básica.

7.5. JUEGO DE REGISTROS DE LA UNIDAD EN COMA FLOTANTE

Esta Unidad se ha rediseñado completamente respecto a la que se usa en el 486. Sin embargo, mantiene compatibilidad 100% binaria con ella. Incorpora un cauce segmentado de instrucciones de ocho etapas, que permite obtener resultados partiendo de instrucciones de coma flotante en cada ciclo de reloj. Las cuatro primeras etapas son las mismas que poseen las unidades de enteros. La quinta y la sexta, corresponden a la ejecución de las instrucciones de coma flotante. La séptima etapa se encarga de escribir el resultado en los registros adecuados y la octava realiza el informe de posibles errores que se hayan producido.

Esta unidad hace uso de nuevos algoritmos que aceleran la ejecución de las operaciones e incluye elementos de hardware dedicados, como son: un multiplicador, un sumador y un divisor. Las instrucciones de suma, multiplicación y carga de datos se ejecutan tres veces más rápido que en un 486.

La Unidad en Coma Flotante o FPU integrada en el Pentium amplía el número de registros que puede manejar el programador de aplicaciones, así como el repertorio de instrucciones. Los nuevos registros que proporciona el coprocesador son ocho generales de 80 bits cada uno y tres de 16 bits, siendo éstos últimos empleados en labores de control y presentación de estado (ver Figura siguiente).

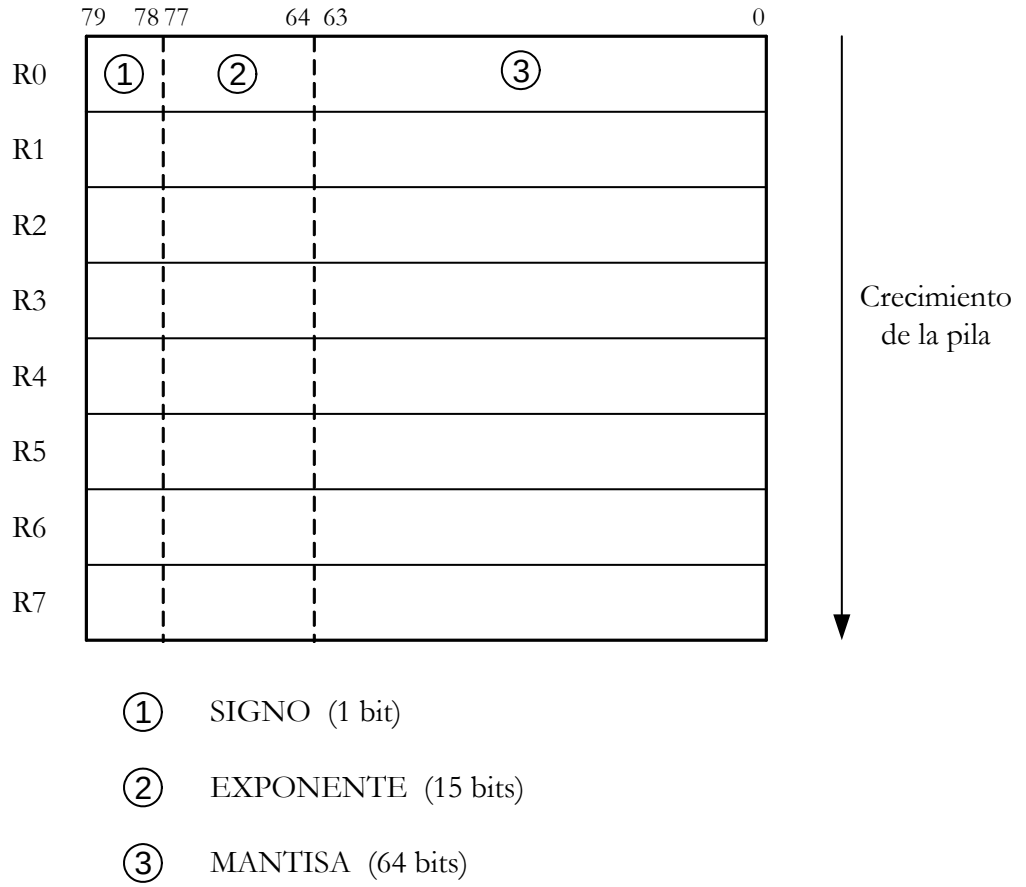

Figura 7.16 – Registros del coprocesador, accesibles al programador de aplicaciones.

Los registros generales R1 a R8 soportan datos con el formato normalizado de doble precisión con 80 bits. Como han de ser manejados en formato de pila, el coprocesador tiene un puntero de control. De pila llamado "St". Toda interacción que tengamos que hacer con los registros del coprocesador se realiza a través del puntero de pila St, donde el último valor introducido es St o St(0) y si hubiéramos rellenado todos los registros el último sería St(7).

EJEMPLO 4

St(0)=1345.
St(0)=5431.

El primer lugar, se carga en el coprocesador el dato 1345 y a continuación el 5431. Por lo tanto, tendremos en St(0) el dato 5431 y en St(1) el 1345.

Podemos observar cómo dichos registros están divididos en tres secciones: signo, exponente de 15 bits y mantisa de 64 bits.

Todas las operaciones del coprocesador se realizan usando los ocho registros generales, que están organizados en forma de pila, no pudiéndose acceder a ellos de forma arbitraria.

El coprocesador soporta los siguientes tipos de datos:

1. Enteros de 16, 32, 64 bits.
2. BCD empaquetados de 80 bits.
3. Números en coma flotante de 32, 64 y 80 bits.

En la figura 7.17 se muestra la distribución de los 16 bits de Palabra de Estado del coprocesador matemático.

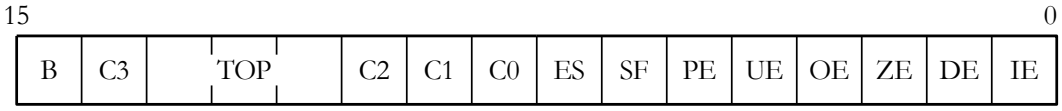

Figura 7.17 – Distribución de los bits en la Palabra de Estado del coprocesador matemático del Pentium

- **TOP**: Este campo de 3 bits de la Palabra de Estado muestra el primer registro. Realiza labores similares al puntero de pila ESP. Estos bits pueden, mediante instrucciones en coma flotante tales como FLD y FSTP (éstas son similares a las instrucciones de tipo entero PUSH y POP), decrementar el TOP en 1 y colocar un valor en su respectivo registro o incrementar el TOP en 1 y retirar el registro pertinente. La pila se incrementa de manera descendente para registrar números menores. Implícitamente, la mayor parte de instrucciones direccionan el registro cima de la pila, es decir, el registro con el número almacenado en el campo TOP del registro de estado. También podremos especificar explícitamente un registro con diversas instrucciones en coma flotante. Fíjese que el registro especificado no es absoluto sino relativo a TOP.

- **B y ES**: Siempre tienen el mismo valor e informa del estado de error. Si ES = 1 se produce una excepción no mascarable cuyo motivo lo indican los bits SP e IE. Bajo ningún concepto el bit B produce ningún dato con respecto al estado de la unidad numérica y en consecuencia al pin BUSY.

- **SF**: Diferencia entre las operaciones inválidas causadas por el desbordamiento de la pila y por otras causas. Si SF está activado el bit C1 diferencia entre un overflow (C1 = 1) y un underflow (C1 = 0). Para una interpretación del código C3-C0 de una comparación u operaciones similares adjuntamos la tabla 7.2 que se muestra a continuación.

INSTRUCCIÓN	C0	C3	C2	C1
FCOM, FCOMP, FCOMPP, FICOM, FICOMP, FTST, FUCOM, FUCOMP, FUCOMPP	Resultados de comparaciones		Operandos no comparables	0 Ó #IS
FCOMI, FCOMIP, FUCOMI, FUCOMIP	No definidas			#IS
FXAM	Clase de operando			SIGNO
FPREM, FPREM1	Q2	Q1	0 = reducción completa 1 = reducción incompleta	Q0 Ó #IS
F2XM1, FADD, FADDP, FBSTP, FCMOVCC, FIADD, FDIV, FDIVP, FDIVR, FDIRVP, FIDIV, FIDIVR, FIMUL, FIST, FISTP, FISUB, FSUBR, FMUL, FPATAN, FRNDINT, FSCALE, FST, FSTP, FSUB, FSUBP, FSUBR, FSUBRP, PSQRT, FYL2XP1	No definido			Roundup ó #IS
FCOS, FSIN, FSINCOS, FPTAN	No definido		1 = operando fuente fuera de rango	Roundup ó #IS (no definido si C2 =1)
FAB, FBLD, FCHS, FDECSTP, FILD, FINCSTP, FLD, CONSTANTES DE CARGA, FSTO (EXT. REAL), FSCH,. FXTRACT	No definido			0 ó #is
FLDENV, FRSTOR	Cada bit se carga de la memoria			
FFREE, FLDCW, FCLEX/FNCLEX, FNOP, FSTCW/FNSTCW, FSTENV/FNSTENV, FSTSW/FNSTSW	No definido			
FINIT/FNINIT, FSAVE/FNSAVE	0	0	0	0

Tabla 7.2 – Códigos en coma flotante para el Pentium.

- Los cuatro **flags de condición de código** (desde C0 a C3) indican el resultado de operaciones aritméticas y de comparación en coma flotante. Estos flags se usan principalmente para el almacenamiento de información usada en excepciones. El flag de condición de estado C1 es usado para gran variedad de funciones. Cuando los bits IE y SF de la CPU están activados, indicando desbordamiento de la excepción overflow o underflow (#IS), el flan C1 lo distingue de la siguiente manera: si está a 1 overflow, sino underflow. Cuando el flag PE de palabra de estado está activado indica que se ha producido un resultado inexacto (redondeo), el flag C1 Es puesto a 1 si los últimos redondeos han sido descendentes. Las instrucciones FXAM activa C1 para examinar el valor del signo.

  El bit de condición de código C2 es usado por las instrucciones FPREM y FPREM1 para indicar una reducción correctamente (o resto parcial). Cuando se completa una reducción completamente, los flags de condición de estado C0, C3 y C1 son activados haciendo referencia a los tres bits menos significativos del cociente (Q2, Q1, y Q0 respectivamente).

  Las instrucciones FPTAN; FSIN, FCOS y FSINCOS ponen el flag C2 a 1 para indicar que el operando fuente está fuera de rango permitido (más/menos 2^{63}).

- **PE**: Precisión.
- **UE**: Underflow.
- **OE**: Overflow.
- **ZE**: División por cero.
- **DE**: Operando desnormalizado.
- **IE**: Operación inválida.

Estos 6 últimos bits (cada uno de ellos por separado) si ponen su bit a 0 es para indicar que no se produce excepción y 1 para indicar que se ha generado la condición de excepción.

Bajo ciertas circunstancias, el Pentium genera una excepción de coprocesador. Estas excepciones pueden ser individualmente enmascaradas. Lo que es más, podemos determinar diferentes modos para precisión y redondeo. La Palabra de Control se emplea para este propósito, y la estructura la mostramos en la figura 7.20:

Figura 7.18 – Distribución de los bits en la Palabra de Control del coprocesador matemático del Pentium

- **IC**: No tiene significado cuando no manejan valores infinitos porque el Pentium aplica el estándar IEEE a las operaciones con coma flotante. En los terrenos de compatibilidad con el 8087 y el 80287 el bit IC está disponible pero no tiene efecto. El Pentium siempre maneja cantidades infinitas en el sentido de más menos infinito., incluso si ponemos IC = 0.

- **RC**: Los dos bits RC controlan el redondeo en el modo definido. En la siguiente tabla se muestran los valores que puede contener el campo RC y el significado de los mismos.

MODO DE REDONDEO	CAMPO RC
Redondeo al más cercano	00B
Redondeo hacia abajo	01B
Redondeo hacia arriba	10B
Redondeo a cero	11B

Tabla 7.3 – Códigos del campo RC para el Pentium

- **PC**: El campo de control de precisión (bits 8 y 9 de la Palabra de Control de la FPU) determina la precisión (64, 53 o 24 bits) de los cálculos en coma flotante realizados por la FPU. Controla el redondeo en las instrucciones ADD, SUB, MUL, DIV y SQRT. En la tabla que se adjunta a continuación, se muestran los valores que puede contener en campo PC y el significado de los mismos.

PRECISIÓN	CAMPO PC
Precisión simple (24 bits)	00B
Reservado	01B
Doble precisión (53 bits)	10B
Precisión extendida (64 bits)	11B

Tabla 7.4 – Códigos del campo PC para el Pentium

- **PM, UM, OM, ZM, DM e IM**: Controlan la generación de una excepción cada uno y su interrupción resultante. Los Pentium provocan seis excepciones diferentes en sus operaciones con coma flotante. Podemos enmascarar las excepciones individualmente empleando los bits PM, UM, OM, ZM, DM e IM. Entonces el Pentium ejecuta una rutina estándar para tratar los respectivos errores utilizando un supuesto estándar. Esto es una parte integral del chip.

Hay disponible un registro de estado más, la Palabra Tag. La FPU usa los valores de los tag para detectar desbordamientos en condiciones de overflow o underflow. Los desbordamientos de overflow ocurren cuando el puntero del campo TOP es decrementado para apuntar a un registro no vacío. Los desbordamientos de underflow se producen cuando el puntero del campo TOP es incrementado para apuntar a un registro vacío, o cuando un registro vacío es referenciado como un operando fuente. Un registro no vacío viene definido como un registro que contiene valor cero (01), valor válido (00) o valor especial (10). Su estructura la mostramos en la figura 7.21 Tag_7 –Tag_0 contiene información, que identifica los contenidos de los ocho registros de datos R7-R0. El coprocesador utiliza esta información para realizar ciertas operaciones a una velocidad superior. Empleando este proceso, el Pentium puede determinar muy rápidamente ciertos valores como NAN, infinito y sin la necesidad de decodificar el valor del registro correspondiente. Mediante las instrucciones de FSTENV/FNSTENV podemos almacenar la Palabra Tag en memoria y examinarla.

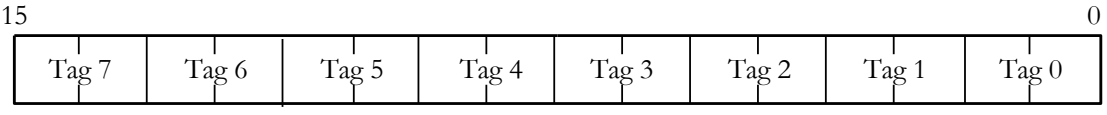

Figura 7.19 – Distribución de los bits en la Palabra Tag del coprocesador matemático del Pentium