PUERTAS DE LLAMADA 12

12.1- TRANSFERENCIAS DE CONTROL

Es manifiesta la necesidad de solicitar algunas rutinas o procedimientos del Sistema Operativo desde las aplicaciones. Sin embargo, no se debe permitir que desde una aplicación se pueda bifurcar directamente a un procedimiento del sistema, porque no todos los procedimientos deben quedar accesibles a las aplicaciones; algunos están reservados exclusivamente al sistema. Además, si se pudiera acceder directamente a cualquier punto de entrada del código privilegiado, el resultado y la seguridad del procedimiento sería aleatorio y se resentiría la seguridad del sistema.

Las transferencias de control son los cambios entre segmentos de código, es decir, el paso de un segmento de código, que se está ejecutando por la CPU, a otro segmento de código. Estos cambios se realizan a través de las instrucciones de salto: JMP y CALL/RET. Estas instrucciones, normalmente, tienen 7 bytes:

```
JMP  CS': DESPLAZAMIENTO
```

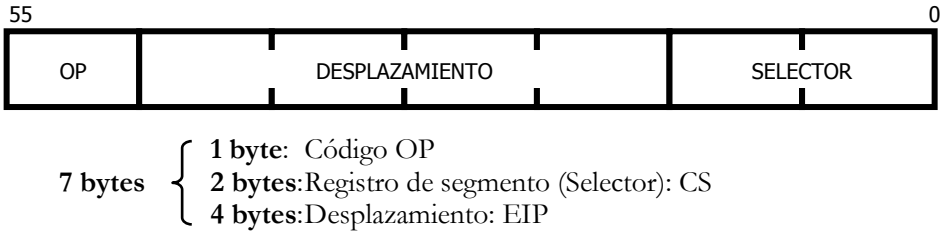

Figura 12.1.Estructura de las instrucciones JMP, CALL y RET.

Las transferencias de control no pueden incumplir las reglas de acceso entre segmentos de código. La regla que forma parte del sistema de protección entre segmentos se encuentra implementada en hardware del Pentium y de otros procesadores de Intel.

Dicha regla establece que sólo se puede acceder a un segmento de código desde otro segmento de código que pertenezca al mismo nivel de privilegio (PL), es decir, sólo podría ser, en principio, posible una transferencia intranivel.

En el Modo Protegido, hay dos tipos de transferencias de control: Intranivel y Internivel.

TRANSFERENCIAS DE CONTROL INTRANIVEL

El segmento peticionario y el segmento solicitado tienen el mismo nivel de privilegio, ya sea dentro de su área local o de la global.

Mientras una tarea esté activa, la CPU maneja sólo dos tablas de descriptores, la GDT y la LDT de dicha tarea. En las transferencias de control intranivel no hay problemas de corrupción, ya que ambos segmentos tienen el mismo nivel de privilegio. Asimismo el Pentium sólo necesita cargar el selector CS para acceder al segmento de código solicitado. En este caso, el procesador sólo comprueba las reglas referidas al tamaño y al tipo:

- que el desplazamiento que implica el valor de EIP no supere el tamaño del segmento, recogido en el campo LIMITE del descriptor, y
- que el bit E del campo TIPO valga 1.

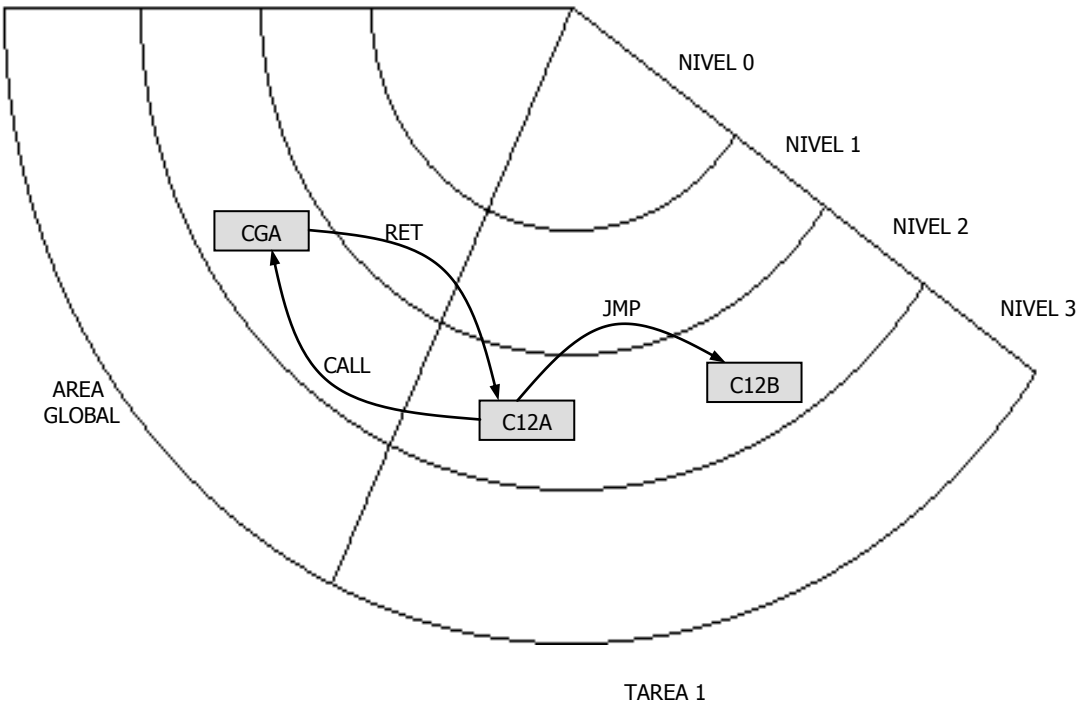

Figura 12.2. Ejemplos de transferencias de control intranivel permitidas.

El campo selector de la figura 12.1 apunta a un descriptor de la tabla GDT o LDT, correspondiente al segmento de código al que se transfiere el control del procesamiento y cuyo DPL debe ser igual al PL del segmento que ha ejecutado la instrucción de transferencia de control (figura 12.2).

Si, al efectuarse la transferencia intranivel, no aparecen errores el contenido del descriptor se carga en el registro caché oculto, asociado al CS.

TRANSFERENCIAS DE CONTROL INTERNIVEL

Se producen cuando el nivel de privilegio del segmento peticionario y el del segmento solicitado son distintos. En principio, es una transferencia de control que no es permitida según las reglas de acceso. Ahora bien, frecuentemente las aplicaciones de bajo nivel de privilegio necesitan con utilizar rutinas del S.O. y para ello se debe acceder a un nivel de privilegio mayor.

Se producen cuando la instrucción de transferencia implica el salto a un código ejecutable de diferente nivel de privilegio que el que hay en curso. Esta posibilidad entraña un alto riesgo de corrupción del sistema y está soportada mediante un recurso denominado Puerta de Llamada, que lo crea el programador del sistema cuando lo juzga conveniente.

La transferencia internivel sólo se puede realizar desde un segmento de código a otro con mayor nivel de privilegio.

En las transferencias de control internivel existe riesgo de corrupción del sistema y para evitarlo, la arquitectura IA-32 tiene grabada en silicio unas reglas que permiten realizar transferencias desde un segmento de código a otro de mayor nivel de privilegio a través de un recurso llamado "Puertas de Llamada" (PLL), que construye a medida el programador de sistemas.

12.2- DEFINICIÓN Y COMPORTAMIENTO DE LAS PUERTAS DE LLAMADA

Para poder acceder desde un segmento de código a otro de mayor nivel de privilegio, los procesadores Pentium utilizan un recurso llamado Puerta de Llamada.

Físicamente, la puerta de llamada consiste en un descriptor local o global, al que se accede a través de una instrucción de formato CALL NOMBRE_PUERTA. El descriptor referenciado por la puerta de llamada permite acceder a un punto de entrada concreto de un segmento de código. La instrucción CALL va asociada con la instrucción RET situada al final del procedimiento accedido. Con RET se devuelve el control al segmento de código original (figura 12.3).

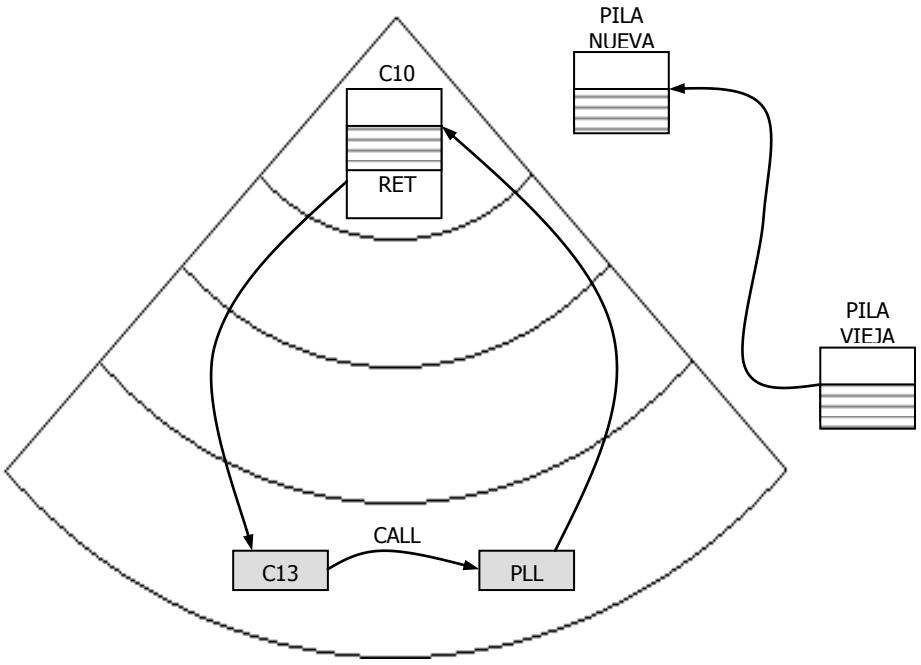

Figura 12.3. Desde un segmento de código ejecutable del nivel 3 se puede acceder aun punto de entrada determinado de un segmento de código del nivel 0, a través de una Puerta de Llamada PLL.

Las Puertas de Llamada, sólo las puede crear el programador de sistemas y es el que otorga esta llave al programador de aplicaciones. Con una Puerta de Llamada se permite acceder a un segmento de código con mayor nivel de privilegio que el del segmento solicitante, pero únicamente por un sitio, es decir, sólo se permite el acceso a un punto de entrada concreto del segmento.

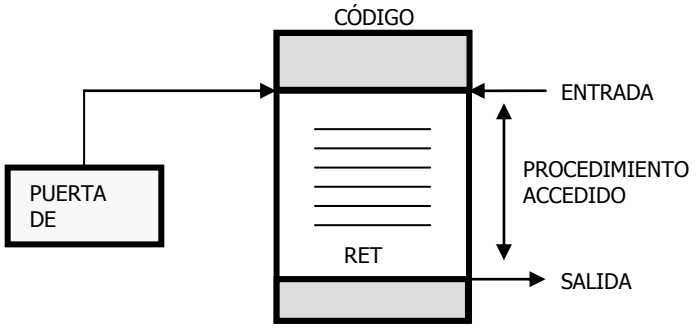

Figura 12.4. Funcionamiento de una puerta de llamada.

En la figura 12.5 se muestra la estructura de un descriptor correspondiente a una puerta de llamada. Obsérvese que en ella, además de los atributos de la puerta, hay un selector y un desplazamiento. Mediante el selector, que se cargará en CS, se determina el nuevo segmento de código, y mediante el desplazamiento se determina el punto de entrada fijo en dicho segmento. El segmento de código peticionario no tiene posibilidad de fijar el punto de entrada, puesto que viene fijado en la propia Puerta de Llamada.

Una Puerta de Llamada (PLL) es un descriptor de 64 bits que puede estar situada en el área global o local, es decir, puede estar en la GDT, o bien, en la LDT.

El descriptor de una Puerta de Llamada tiene básicamente las siguientes funciones:

1. Especifica el segmento de código a acceder.
2. Define un punto de entrada fijo a una rutina determinada del segmento de código al que se accede.
3. Determina el número de parámetros que se copiarán automáticamente de la pila del peticionario a la pila del segmento solicitado.
4. Define los atributos de la Puerta de Llamada.

La estructura del descriptor de una Puerta de Llamada es la siguiente:

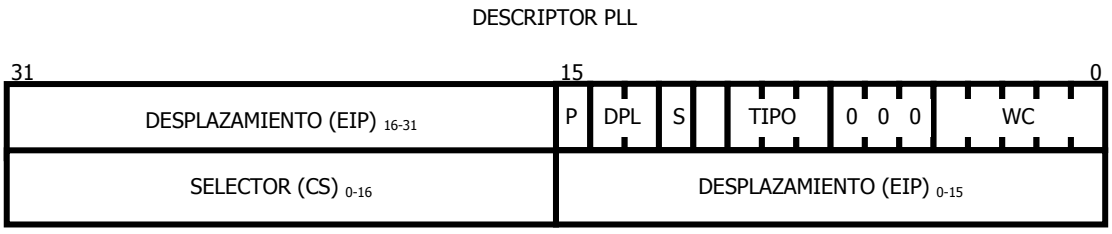

Figura 12.5. Estructura de un descriptor correspondiente a una puerta de llamada. El campo TIPO, para el Pentium, tiene el valor C en hexadecimal.

- **Selector:** El Selector es un valor de 16 bits que se carga en CS y determina el nuevo selector del segmento de código.

- **Desplazamiento:** Determina el punto de entrada fijo en el segmento. Dicha entrada es, generalmente, la primera instrucción de una rutina.

- **Atributos:**

  - **P**: BIT DE PRESENCIA: Indica si el segmento al que referencia el descriptor está cargado en la memoria principal (P=1), o bien está ausente (P=0). Es decir, si dicho descriptor es válido o no.
  - **DPL**: NIVEL DE PRIVILEGIO DEL DESCRIPTOR: Indica el nivel de privilegio de la PLL a la que referencia.
  - **S**: SEGMENTO DEL SISTEMA: Si S=0 indica que se trata de un segmento del sistema. Por el contrario si S=1 se trata de un segmento de código.
  - **TIPO**: Con 4 bits (1100) indica que se trata de un descriptor de Puerta de Llamada. Para el Pentium tiene el valor C en hexadecimal.
  - **WC**: CONTADOR DE PALABRAS: Son 5 bits que indican el número de parámetros que se transfieren desde la pila del nivel del segmento peticionario a la del segmento solicitado. El número máximo de parámetros que se pueden pasar es de 32 dobles palabras (2^5).

Debido a la compatibilidad descendente de todos los procesadores Intel, el Pentium tiene la posibilidad de usar Puertas de Llamada del 80286, que se diferencian en el formato descriptor: por: El campo de $DESPLAZAMIENTO_{16\text{-}31}$ que está reservado, es decir, sus bits están puestos a 0. Además, del campo WC, que indica el número de palabras de 16 bits, no de palabras dobles de 32 bits, que se transfieren como parámetros a través de la puerta de llamada.

FASES DE UNA TRANSFERENCIA POR PLL

La operatividad de una Puerta de Llamada, viene expresada de forma gráfica en la figura 12.6 y consta de las siguientes fases:

- **1ª FASE:** En el programa en curso aparece una instrucción de llamada a una puerta: CALL PLL.

- **2ª FASE:** Se accede al descriptor de la puerta de llamada, en donde se obtiene, junto con los atributos de la puerta, un selector y un desplazamiento.

- **3ª FASE**: El valor del selector de la puerta se carga en CS, con el que se accede a un descriptor de segmento de código cuyo contenido (base, límite y atributos) se carga en el registro caché oculto asociado a CS.

- **4ª FASE:** Mediante los valores de la base y el límite situados en el registro caché de CS se determina el segmento a acceder. Luego la CPU comienza la ejecución de la instrucción situada en la dirección obtenida al sumar a la base el desplazamiento existente en la Puerta de Llamada.

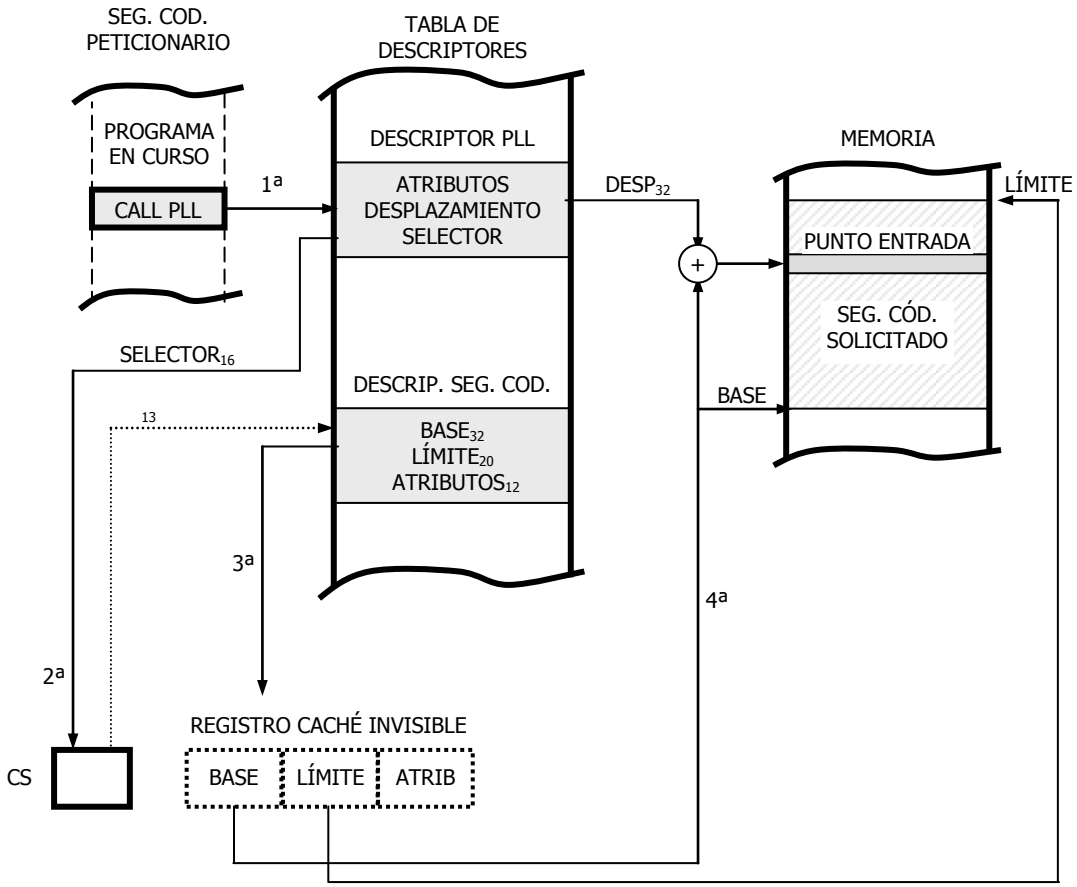

Figura 12.6. Mecanismo para la transferencia de control internivel, utilizando una Puerta de Llamada, PLL.

El acceso a un segmento de código mediante una Puerta de Llamada supone la comprobación de los siguientes niveles de privilegio:

- CPL: nivel de privilegio del segmento de código en curso.
- RPL: nivel de privilegio del segmento peticionario.
- DPL: nivel de privilegio del descriptor de la puerta de llamada.
- DPL del descriptor del segmento solicitado.

También es verificado el bit C (Conforming) del segmento de código destino.

REGLAS DE MANEJO DE LAS PLL

En el manejo de las Puertas de Llamada se debe cumplir tres reglas:

1. Para acceder a una Puerta de Llamada desde un segmento de código, el nivel de privilegio de este último ha de ser igual o mayor que el de la puerta. Esta regla es similar a la expuesta para realizar el acceso a los segmentos de datos.

2. Mediante una Puerta de Llamada sólo se puede transferir el flujo de control a un segmento de código de mayor nivel de privilegio que el peticionario.

3. Cuando se ejecuta una instrucción CALL PLL se guarda en la pila del nivel correspondiente a la PLL (pila vieja) la dirección de retorno CS:EIP al segmento

4. peticionario. También se guardan los parámetros que referencian a la pila vieja SS:ESP y los que indica el WL. Asimismo se realiza una transferencia de datos de la pila vieja a la nueva, es decir, la correspondiente al PL del segmento destino. Efectuada la transferencia a un segmento de código con mayor PL, el retorno al segmento de código primitivo se realiza con la instrucción RET, que recupera de la pila el selector y el desplazamiento iniciales. El retorno sólo se puede llevar a cabo a un segmento con igual o menor PL.

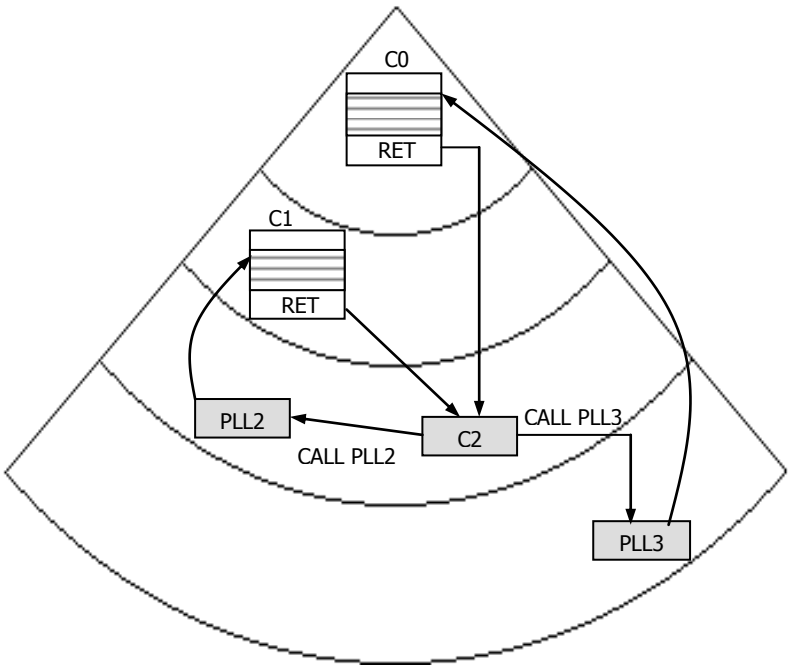

Figura 12.7. Posibles ejemplos de transferencia de control internivel, usando Puertas de Llamada.

Las Puertas de Llamada proporcionan una rápida entrada a puntos concretos del sistema operativo, considerando a éste último como parte de una tarea.

12.3-COMPORTAMIENTO DE LA PILA EN LAS TRANSFERENCIAS INTERNIVEL

En cada PL en donde existe un segmento de código debe existir también un segmento de pila.

Si se compartiese el mismo segmento de pila en una tarea, o sea, la utilizasen todos los segmentos de código situados en los diferentes niveles de privilegio, la pila del sistema operativo podría ser corrompida por el código de aplicación de la tarea.

La regla de acceso a los segmentos de pila exige que el nivel de privilegio del segmento de código coincida con el de la pila.

Cada tarea dispone de un segmento de pila independiente para cada nivel de privilegio en el que residan segmentos de código. Cuando se cambia de nivel de privilegio, se cambia automáticamente de pila, modificándose el contenido de SS y ESP. Este es un recurso suplementario de protección, que evita la mezcla de parámetros procedentes de diversos PL en la pila (figura 12.8).

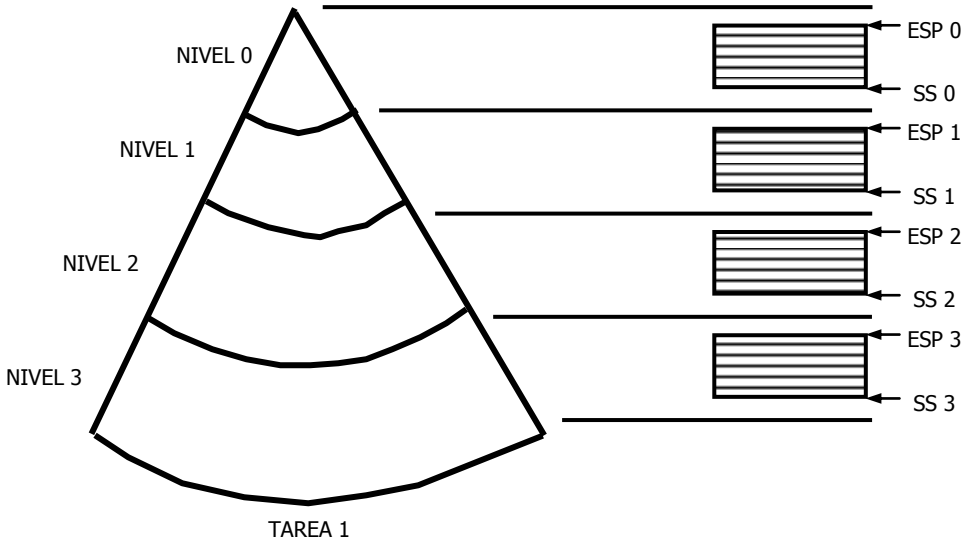

Figura 12.8. Cada tarea dispone de un segmento de pila para cada nivel de privilegio en el que exista código ejecutable.

Los parámetros que definen a los segmentos de pila correspondientes a los diversos niveles de privilegio se hallan guardados en el Segmento de Estado de la Tarea, TSS.

En cuanto se produce una transferencia de control mediante una Puerta de Llamada el procesador realiza un traspaso de parámetros a la pila del segmento accedido. El cambio de pila se ha de realizar para prevenir la mezcla de datos de diferente PL entre rutinas de diferentes niveles de privilegio en el caso en que compartieran la misma pila.

Cuando se produce una transferencia de control internivel, es muy frecuente tener que traspasar o copiar un conjunto de parámetros desde la pila primitiva a la nueva. Mediante el uso de Puertas de Llamada se consigue una autocopia automática de parámetros entre las dos pilas afectadas.

Apréciese en la figura 12.5, que en la estructura del descriptor de puerta de llamada hay un campo de cinco bits denominado Contador de Palabras (WC: Word Counter). En dicho campo se

carga un valor que especifica el número de palabras (Dword), con un máximo de 31, que se van a transferir, desde la cima de la pila en curso a la nueva, situada en otro PL.

En el caso en que se necesite transferir más de 31 parámetros a la rutina destino, uno de dichos parámetros puede ser un puntero a un bloque de datos en memoria.

En realidad, además de copiarse en la pila nueva el número de palabras de la pila vieja indicado en WC, también se salvan:

A) El SS y el ESP de la pila en curso.
B) La dirección de retorno (CS y EIP) del segmento de código peticionario.

En la figura 12.9 se presenta la estructura del segmento de la pila nueva, al que se ha pasado mediante el uso de una puerta de llamada.

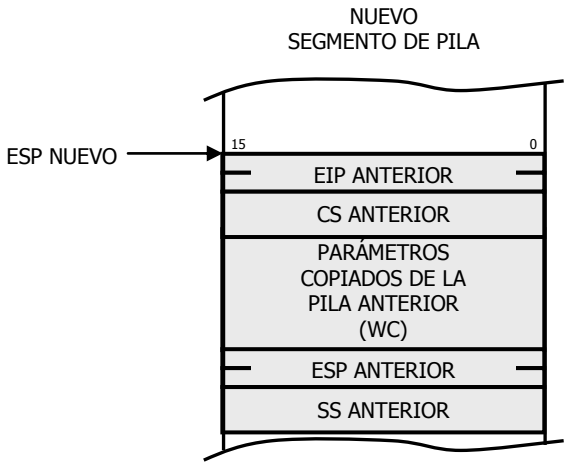

Figura 12.9. Estructura del contenido que se salva en el segmento de pila, al cual se ha accedido por una Puerta de Llamada.

Debe ser el programador de sistemas el encargado de confeccionar las rutinas o procedimientos que pueden ser usados por los módulos de aplicación, así como diseñar las Puertas de Llamada necesarias para acceder a ellas.

La Puerta de Llamada es el mecanismo más seguro y flexible para propiciar a las aplicaciones el acceso al sistema por un punto concreto sin riesgo de degradar los niveles de seguridad. La puerta de llamada puede residir en el espacio local o en el global y, según la diseñe el programador de sistemas, conseguirá:

a) Determinar los módulos de las aplicaciones capaces de acceder a las rutinas del sistema. Para ello debe situar a la PLL en el nivel de privilegio y espacio adecuados.

b) Especificar el punto concreto por el que se permite la entrada a una rutina del sistema, estableciendo el valor del desplazamiento en el descriptor de la PLL.

12.4- EL ESCENARIO DEL CABALLO DE TROYA

Un caballo de Troya es un programa destructivo que aparenta ser legítimo y que con frecuencia se adjuntan en correos, en programas o en juegos de software gratis. Los caballos de Troya pueden encontrar información de contraseñas, hacer a los sistemas más vulnerables a invasiones o sabotear datos en el disco duro de un usuario.

El escenario del caballo de Troya se origina cuando se envía a través de la Puerta de Llamada parámetros prohibidos, de forma que la rutina del nivel 0 permite el acceso a segmentos de datos a los que el segmento peticionario desde su PL no puede acceder.

Las Puertas de Llamada, como se ve, plantean un grave problema al sistema de protección, puesto que su empleo permite aumentar el nivel de privilegio del segmento de código en ejecución (peticionario). Hay que evitar que desde un segmento de código se pueda acceder a un segmento de datos de mayor nivel de privilegio, hecho factible cuando, como paso intermedio, se sube el nivel del código usando una Puerta de Llamada.

Supóngase el caso mostrado en la figura 12.10:

Se ve que hay un segmento de código, perteneciente a una aplicación con PL=3, que a través del mecanismo de transferencia de la Puerta de Llamada PLL, permite acceder al segmento de código del S.O. con PL=0. Un programador podría enviar por medio de la transferencia de parámetros a la pila, utilizando WC (Contador de Palabras), el selector del segmento de datos del sistema y algún dato para escribir en él, con lo que se estaría violando la seguridad del sistema.

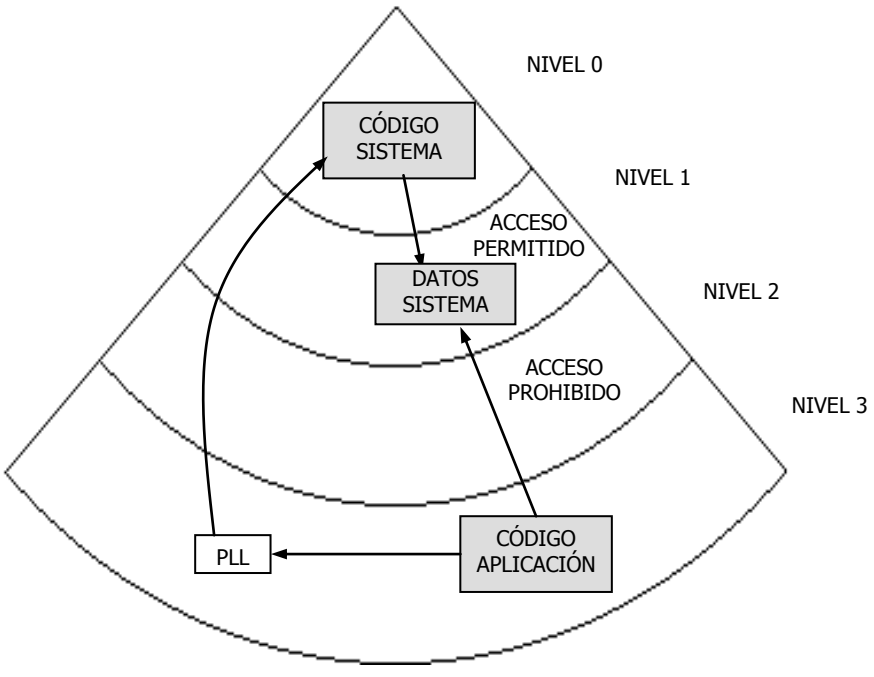

Figura 12.10. Hay que evitar que el segmento "Código Aplicación" pueda acceder en escritura al segmento "Datos Sistema", a través de una Puerta de Llamada.

Para evitar el escenario del caballo de Troya, existe dos recursos fundamentales que se describen en apartados posteriores. Se trata de los "segmentos ajustables" y "la instrucción ARPL".

Si en el establecimiento de las reglas de acceso sólo se comparase el CPL (Current Privilege Level) del segmento de código peticionario con el DPL (Data Privilege Level) del segmento a acceder, podría reconstruirse el escenario del caballo de Troya, que ya se ha explicado.

Para realizarlo se eleva el nivel de privilegio mediante una Puerta de Llamada y se pasan a la pila del nuevo nivel un parámetro, que posteriormente se utiliza como selector, para acceder a un segmento de datos prohibido para el segmento de código inicial, es decir peticionario. Para evitar esta posible transgresión, la comparación del CPL del segmento a acceder se lleva a cabo con el Nivel de Privilegio Efectivo, EPL, cuyo valor es:

$$EPL = Máx\ (CPL,\ RPL)$$

Recuérdese que el campo RPL (Request Privilege Level) ocupa los dos bits de menos peso del selector cargado en el registro de segmento. El RPL es el PL del segmento de código que inicialmente se había solicitado al selector. Es una especie de recordatorio del CPL peticionario del selector. En el caso que se comenta, el valor del RPL del selector de la Puerta de Llamada tendrá de valor tres, al ser éste el CPL del segmento Código Aplicación. Por lo tanto, aunque se accede al segmento "Código sistema" de CPL=0, el EPL resultante será el máximo valor numérico entre RPL y CPL, o sea, tres. Así se evita el acceso al segmento de Datos Sistema con DPL=1.

Con el empleo del EPL se tiene en cuenta el nivel de privilegio del segmento que llama a la Puerta de Llamada, evitando que ningún elemento pueda aumentar su nivel de privilegio, de forma indirecta.

Las instrucciones de retorno RET, sólo pueden disminuir el nivel de privilegio en la transferencia de control y durante su ejecución se examinan los registros de segmento de datos y se ajusta el nivel de privilegio al del código al que ha retornado.

12.4.1- Segmentos ajustables

Reciben el nombre de segmentos ajustables, aquellos segmentos de código que tienen el bit C (conforming o ajustable) a 1. Su característica fundamental es que pueden ser llamados directamente desde otro segmento de código con igual o menor nivel, ejecutándose con el mismo nivel que el del segmento peticionario.

Un segmento ajustable ajusta su PL al del segmento de código desde el que se transfiere el control de la CPU.

Mediante los segmentos ajustables, muchas funciones generales, como transformaciones matemáticas, pueden ser accedidas desde cualquier nivel de privilegio directamente (sin Puerta de Llamada) y ser ejecutados con el mismo nivel que el del segmento que lo ha solicitado, evitando que se reproduzca el escenario del caballo de Troya (figura 12.11).

Todos los segmentos que tiene el bit C=1, al ser accedidos desde otro segmento de código, cambian su PL a un nivel que se llama "efectivo" (EPL). El EPL es igual al mayor numérico entre el nivel del segmento peticionario y el nivel del segmento solicitado.

$$EPL = (CPL,\ RPL)_{\text{MAYOR NUMÉRICO}}$$

El problema de hacer ajustables los segmentos es la degeneración del nivel de privilegio dando lugar a que, durante el intervalo de tiempo en que la puerta de llamada esté activa, el segmento de código degenerado de nivel pueda quedar corrompido.

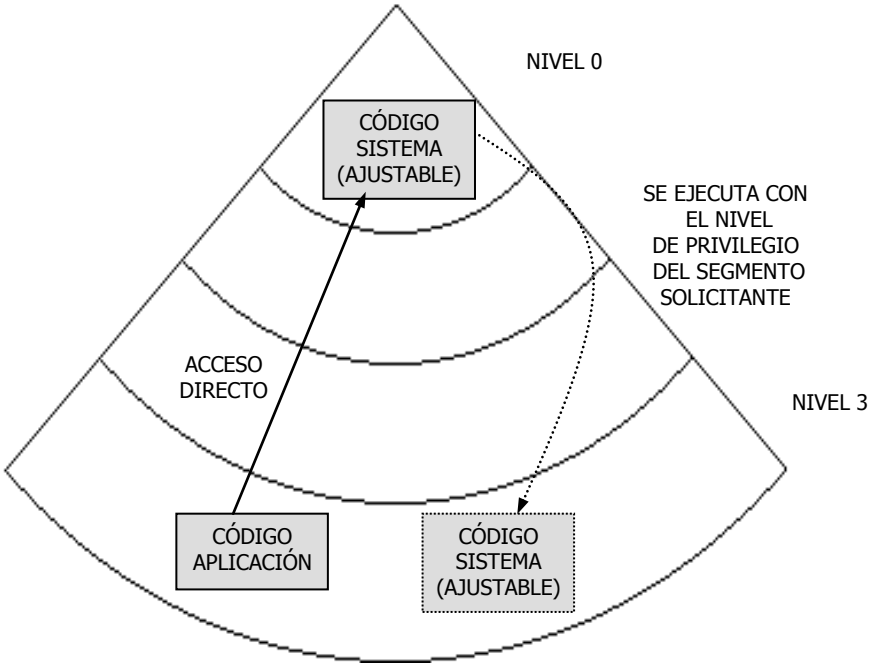

Figura 12.11. Cuando un segmento es ajustable, se ejecuta con el mismo nivel de privilegio que el que tiene el segmento peticionario.

Para transformar un segmento de código en ajustable, basta con poner a 1 el bit C ("Conforming") de los atributos de su descriptor, como se indica en la figura 12.12.

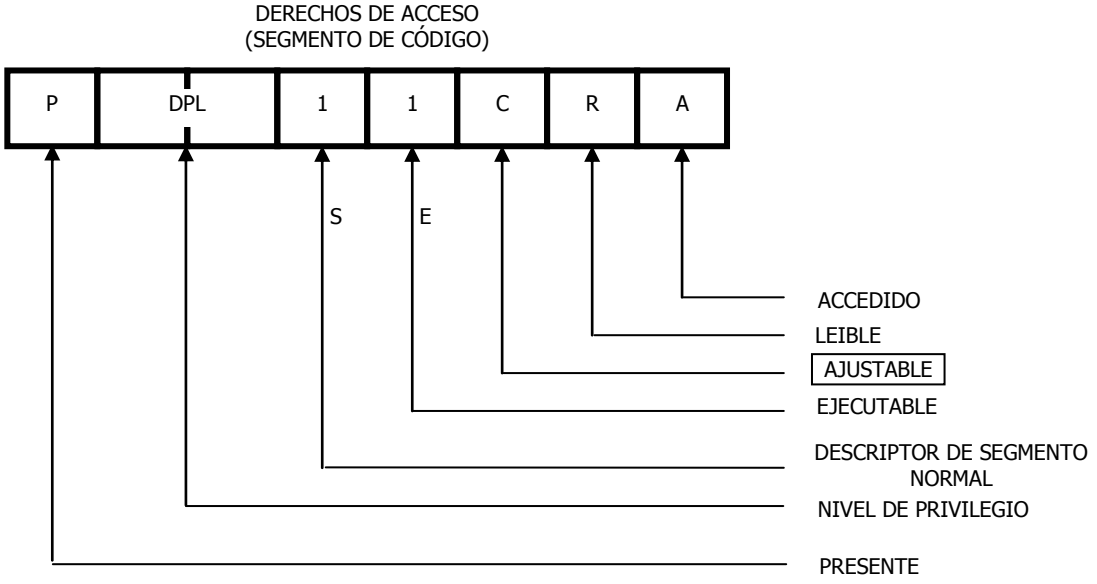

Figura 12.12. El bit C de los derechos de acceso de un descriptor de segmento de código, indica si dicho segmento es ajustable (C=1).

12.4.2- Instrucción ARPL

La instrucción ARPL (Adjust Requested Privilege Level) sirve para cargar en el campo RPL de un selector, el valor EPL, que es el máximo valor numérico entre el CPL peticionario y el RPL del selector a ajustar.

```
EPL= Máx (CPL, RPL).
```

Es una instrucción muy eficaz para aquellos casos en los que un mismo selector se utiliza en diversos niveles de privilegio, evitando que se pueda acceder a segmentos de datos prohibidos.

El formato de la instrucción es:

```
ARPL (SELECTOR A AJUSTAR),(SELECTOR CS DEL PETICIONARIO)
```

Los dos selectores que manipula ARPL están ubicados en los registros de la CPU, por ejemplo, ARPL CX, DX.

En la figura 12.13 se muestra el contenido de CS y DX antes de realizar la instrucción y la forma en que se altera el contenido de CX, en donde los dos bits que conforman su campo RPL, toman el valor del EPL calculado en la instrucción.

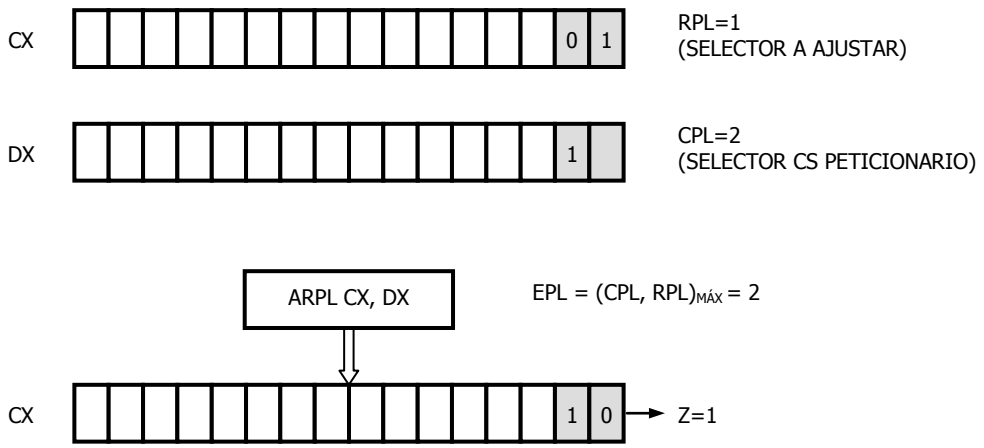

Figura 12.13. Comportamiento de la instrucción ARPL que modifica el valor RPL del selector a ajustar, con el del EPL calculado.

Cuando al ejecutase la instrucción ARPL, se produce un cambio en el campo RPL del selector a ajustar, el señalizador Z pasa a valer 1, lo que indica que el RPL del selector a ajustar es menor que el CPL del selector CS peticionario, de esta forma el sistema se da cuenta de que se quería hacer trampa.

Siempre que sea llamado un procedimiento, el CPL del segmento que llama se almacenará en el campo RPL del selector CS de la dirección de retorno (figura 12.14).

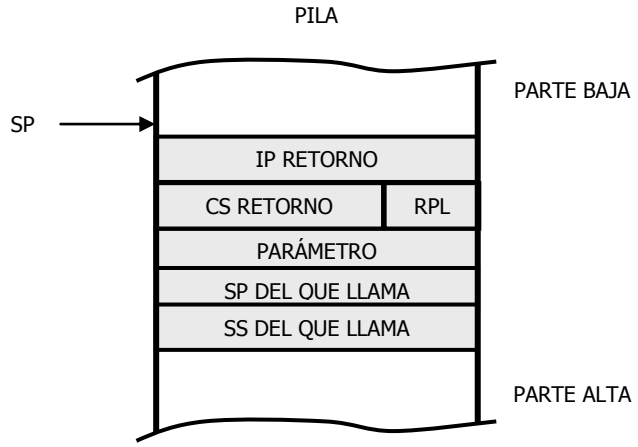

Figura 12.14. El CPL del segmento que llama es salvado en el campo RPL del selector CS del retorno.

Se propone un ejemplo de aplicación de la instrucción ARPL en la posible validación de la carga de un selector en un segmento de datos DS en donde se manejan parámetros de 16 bits. Véase la figura 12.15.

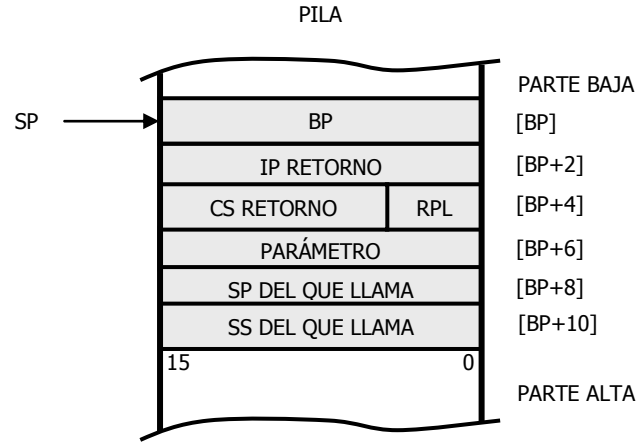

Figura 12.15. Estructura de la pila cuando se emplea la instrucción ARPL para la validación.

Obsérvese en el programa confeccionado en lenguaje Ensamblador, cómo se trata de cargar un parámetro pasado por la Puerta de Llamada correspondiente a un segmento de datos que se quiere cargar en DS. La instrucción ARPL previa evita que se plantee el escenario del caballo de Troya.

```
PROCED_A    PROC  FAR     WC (1)
                                    PUSH        BP
                                    MOV         BP, SP
                                    MOV         AX, (BP+6)
                                    ARPL   AX, (BP+4)
                                    MOV         DS, AX
(si Z=1, ha cambiado el PL)
                                    -------
            -------
            POPBP
            RET2
PROCED_A    ENDP
```

12.5- DESCRIPTORES ALIAS

Las tablas de descriptores GDT, LDT, IDT, etc, son consideradas por el sistema de protección del Pentium como si se tratase de segmentos de código, lo que significa que su contenido no se puede modificar, pues en ese tipo de segmento siempre está prohibida la escritura.

El S.O. debe actualizar los contenidos de los descriptores de forma que, si un descriptor marca como "no presente" al segmento que referencia, cuando se direcciona, el S.O. lo carga en la memoria principal y, luego, deberá actualizar el bit P poniéndolo a 1; además, debe introducir en los campos base, límite y atributos los valores correspondientes a la zona de memoria usada. Esto significa que el S.O. debe escribir en la tabla de descriptores.

También al bit A (Accedido) del campo de atributos del descriptor debe ponerlo a 0, periódicamente, el S.O., para controlar el número de veces que ha sido usado y poder aplicar los algoritmos de intercambio de segmentos.

Para superar la prohibición de escritura en las tablas de descriptores, el S.O. crea dinámicamente descriptores alias, que son "copias" de los segmentos para los que se desean modificar sus derechos de acceso. El descriptor alias asigna al segmento el carácter de "datos", para poder escribir en ellos. Para limitar el acceso a estos descriptores al S.O., se les sitúa en el nivel de privilegio 0 (figura 12.16).

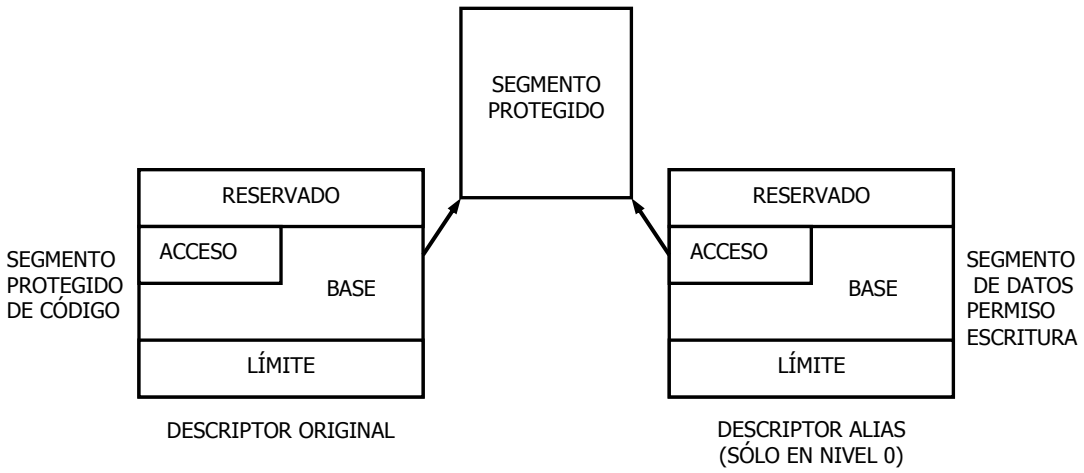

Figura 12.16. El S.O. puede crear descriptores alias, que dan al segmento que referencian el carácter de segmento de datos, en los que es posible escribir y modificar su contenido.

El empleo de los descriptores alias permite el manejo de un segmento como si fuese de código contemplado desde la aplicación, mientras que desde el S.O. se manipula como si se tratase de un segmento de datos (figura 12.17).

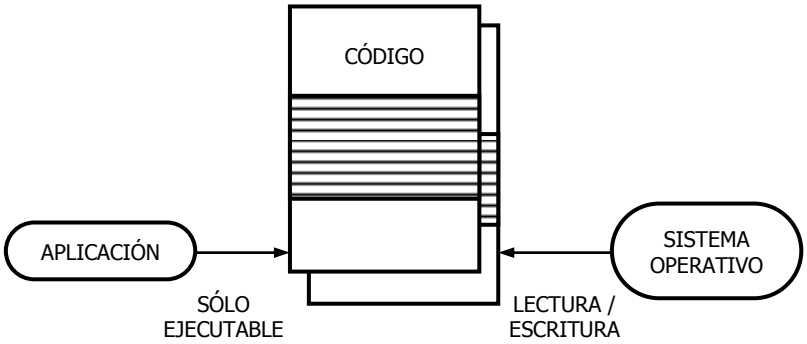

Figura 12.17. Utilización del descriptor alias.

También es posible asignar derechos de acceso diferentes a distintos usuarios empleando este tipo de descriptor (figura 12.18).

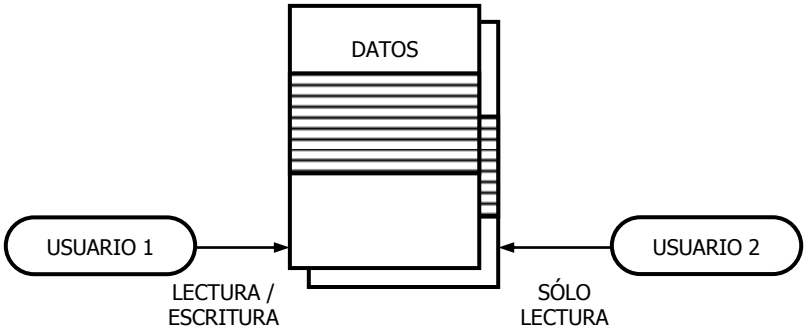

Figura 12.18. Según el usuario que maneja el segmento, cambian los derechos de acceso.

Finalmente, otra interesante aplicación de los descriptores alias es cuando se colocan en la GDT y permiten acceder a todos los elementos que referencian desde cualquier parte del sistema (figura 12.19).

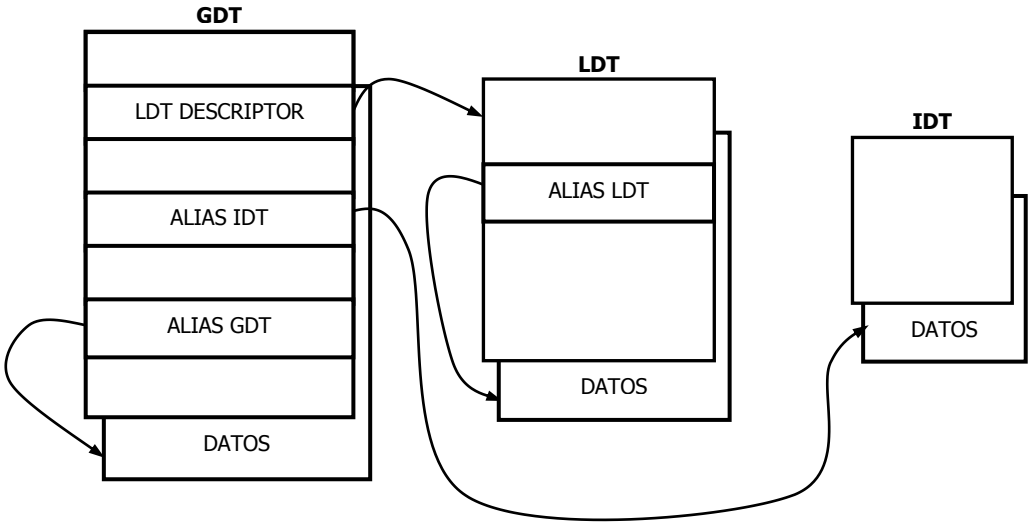

Figura 12.19. Los descriptores alias en la GDT son siempre accesibles.

12.6- PARTICULARIDADES DE LOS SEGMENTOS DE PILA

La pila es una zona cualquiera de la memoria, con estructura LIFO. El control de la pila es soportado directamente por el procesador y se emplea para:

- Almacenar la dirección de retorno en subrutinas e interrupciones.
- Espacio de trabajo de un procedimiento.
- Para paso de parámetros.

El mecanismo de direccionamiento de la pila se muestra en la figura 12.20, donde se aprecia que el puntero ESP, crece hacia la base del segmento de pila direccionada por SS. Se trata de un crecimiento hacia abajo, es decir, hacia direcciones inferiores.

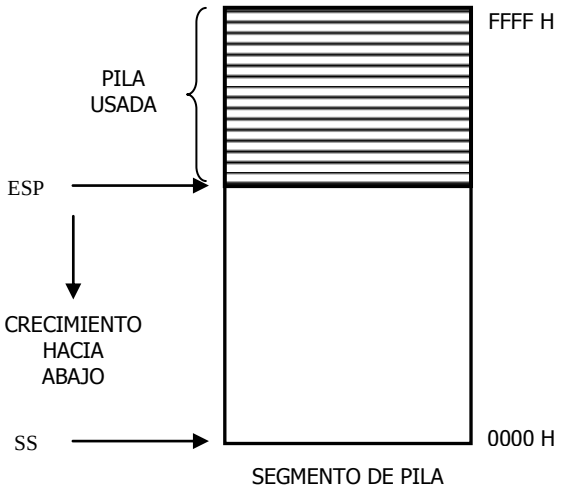

Figura 12.20. El ESP crece hacia el SS, o sea, la pila va creciendo hacia direcciones numéricamente inferiores o decrecientes.

La manipulación de la pila puede dar lugar a dos tipos de fallos o errores:

1. **Desbordamiento positivo de la pila**.
   Cuando se excede el límite máximo asignado al segmento de pila.

2. **Desbordamiento negativo de la pila.**
   Suele consistir en un error de programa, en el que se intercambian instrucciones PUSH por POP, dando lugar a que la pila se desborde por el límite opuesto al caso anterior.

Cualquiera de los dos errores típicos de pila compromete la integridad del sistema.

El vector de excepción número 12 se dedica a resolver los fallos provenientes del manejo de la pila. En caso de desbordamiento, la rutina de excepción puede encargarse de aumentar el límite del segmento. También se utiliza este vector cuando el segmento no está presente. En ambas situaciones, se salva en la pila de la rutina de excepción el selector del código de error, que contiene el segmento que ha producido el fallo.

En la figura 12.21 se muestra la diferencia fundamental entre el funcionamiento de un segmento de pila convencional y otro con expansión hacia abajo (expand down). En este último caso, detrás del límite del segmento de pila existe una zona de memoria en la que es posible el crecimiento de la pila sin necesidad de modificar el valor de SS.

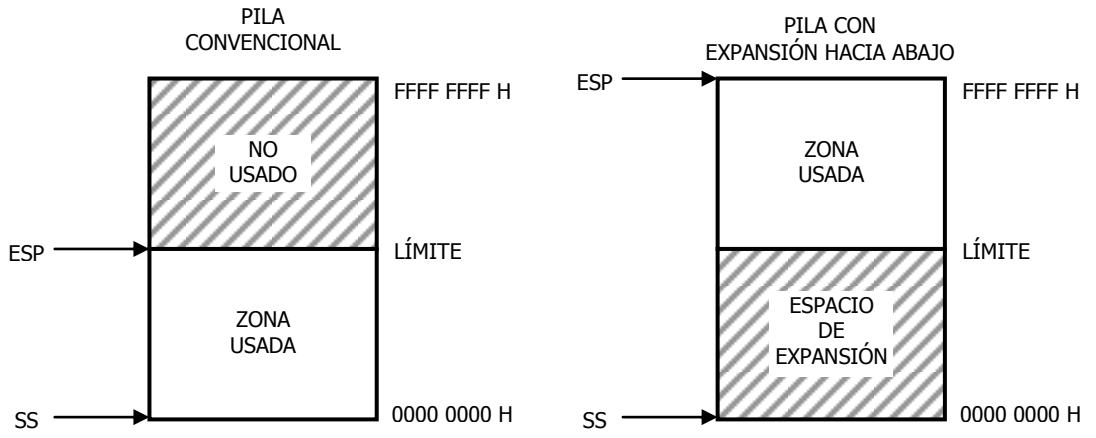

Figura 12.21. Diferencia entre el comportamiento de una pila convencional y otra con expansión hacia abajo.

En los atributos de los descriptores de segmentos de datos existe un bit DE (Expand Down), que indica si el crecimiento de las direcciones implica el aumento de su valor (ED=0), o por el contrario, su disminución (ED=1).

Si ED=0, se debe cumplir que el DESPLAZAMIENTO ha de ser igual o menor que el LIMITE.

Si ED=1, el desplazamiento puede tener un valor mayor que el límite contenido en el descriptor.

Cuando se trata de un segmento de datos, con ED=1, el S.O. sólo debe cambiar el valor del límite (actualizando el descriptor) para ampliar la capacidad de la pila. Los desplazamientos no se alteran como consecuencia de dicha ampliación. Con esta posibilidad, es posible usar pilas pequeñas pero ampliables cuando se haga necesario (figura 12.22).

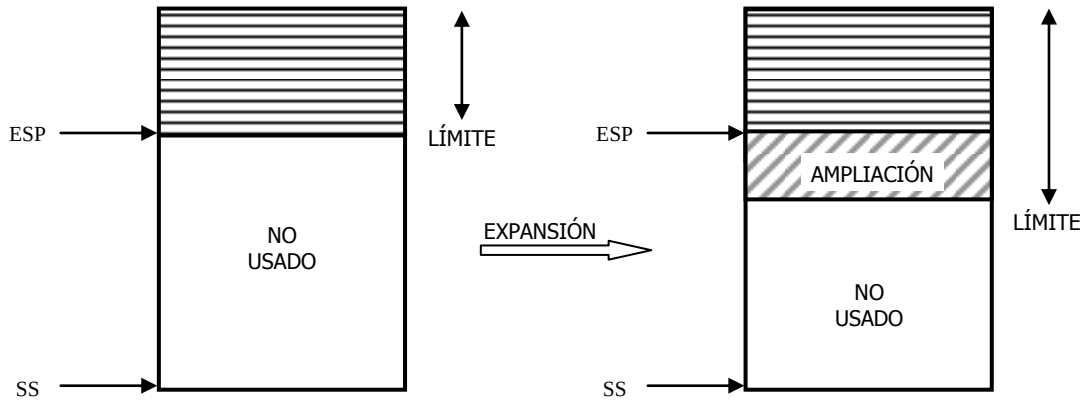

Figura 12.22. El sistema operativo sólo debe cambiar el valor del límite cuando se desea ampliar la capacidad de la pila.