# ANEXO B

# EJERCICIOS

# 1. TEMA 8: SEGMENTACIÓN

## 1.1. Ejercicio 1

**En un entorno multitarea hay tres tareas en memoria (T1, T2 y T3), estando activa la tarea T2. En un instante dado, genera la dirección virtual (en binario):**

00 0000 0000 0100 0000 0000 0000 0000 0000 0000 0000 0111

**Sabiendo que la situación de la tabla de descriptores global y de la propia de la tarea, estando los valores expresados en hexadecimal, es la indicada en la figura 26.1, responder a las siguientes preguntas:**

**a) ¿A qué descriptor de segmento se accede?**
**b) ¿Dónde se encuentra el segmento accedido? Realizar un diagrama de la memoria que muestre claramente dónde se encuentra.**
**c) ¿Cuál es la dirección de la base del segmento?**
**d) ¿Cuál es el tamaño del segmento?**
**e) ¿Está presente el segmento en memoria?**
**f) ¿De qué tipo de segmento se trata?**
**g) ¿Cuál es la dirección física a la que se accede?**

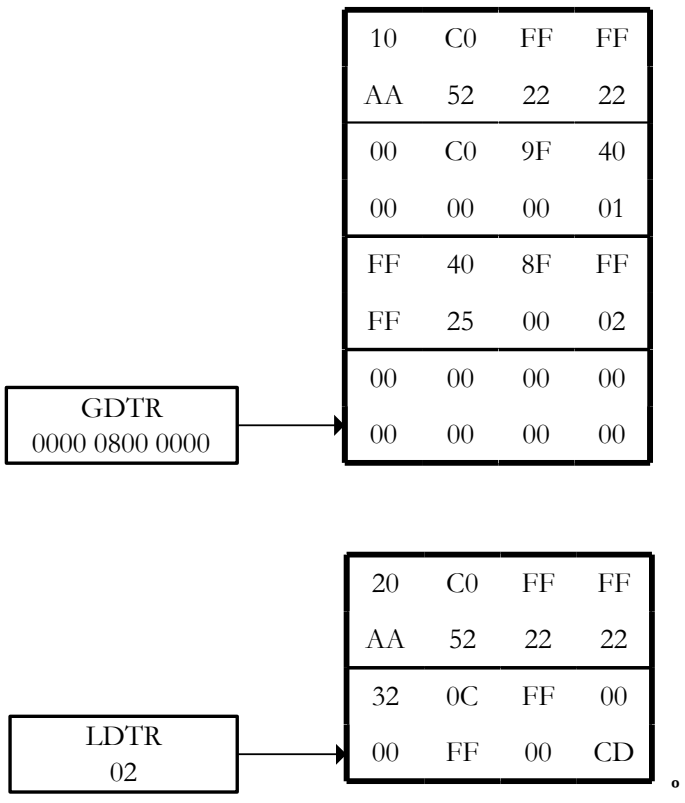



Figura 1 - Contenido de las tablas de descriptores

## Resolución

**a) ¿A qué descriptor de segmento se accede?**

Lo primero que necesitamos saber es a qué descriptor se accede. Para ello, tomamos los primeros 16 bits de la dirección virtual, como vemos en la figura 2. De esos 16 primeros bits, los dos de menos peso serán el CPL (nivel de privilegio), el siguiente será el bit TI, que indicará si la tabla a la que se accede es la GDT o la LDT de la tarea en curso, y los 14 restantes serán el selector de segmento, que indicará a que selector se accede dentro de la tabla que corresponda.

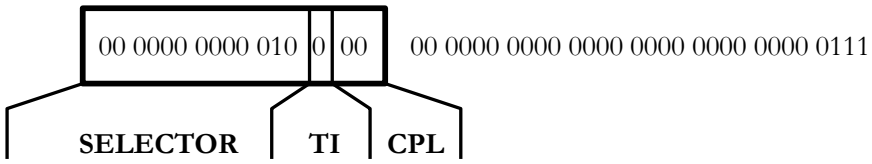



Figura 2 - Separación de los 16 bits de más peso de la dirección virtual

Como TI = 0, se utiliza la GDT. Vemos que el valor del campo Selector es 1, por lo que se accede al segundo descriptor de la GDT (recordemos que el primero es el selector nulo y no se utiliza).El selector accedido es por tanto, el que se indica en la figura 3

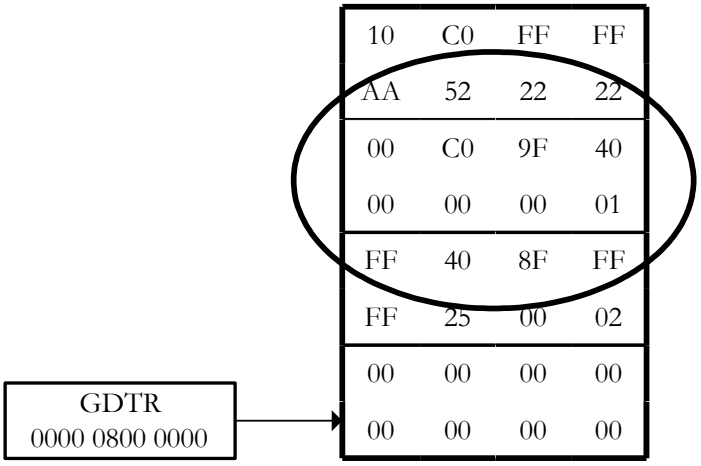



Figura 3 - Selector al que se accede

**b) ¿Dónde se encuentra el segmento accedido? Realizar un diagrama de la memoria que muestre claramente dónde se encuentra.**

Dado que su descriptor se encuentra en la GDT, el segmento se estará en el área global de memoria. Para saber en qué nivel de privilegio, es necesario examinar los atributos del descriptor. Para eso es conveniente recordar cuál es la estructura de un descriptor de segmento (figura 4), lo que nos va a permitir ubicarlos fácilmente.

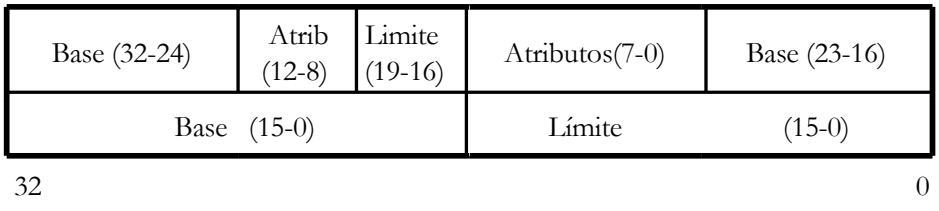



Figura 4 - Estructura de un descriptor de segmento

Examinamos ahora los doce bits que indican los atributos.

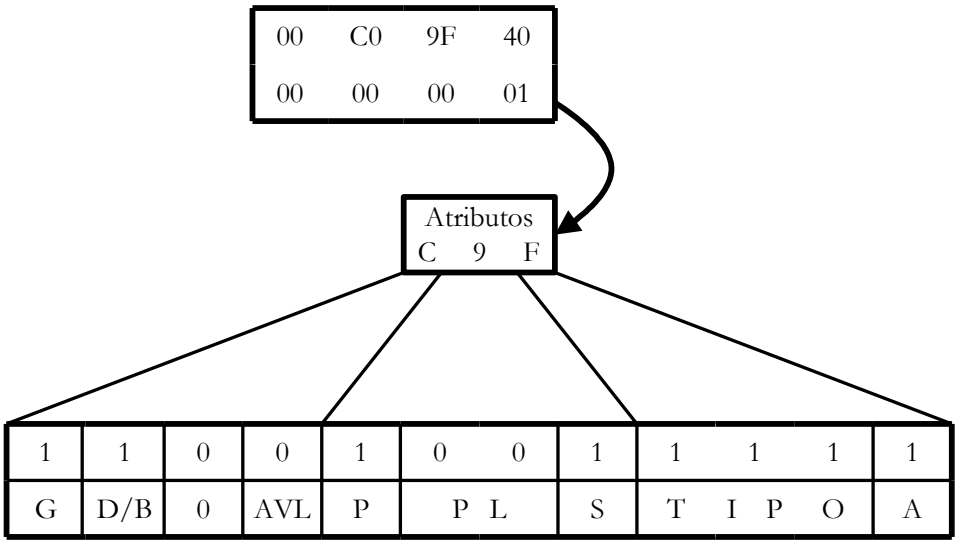



Figura 5 - Atributos de un descriptor

Vemos que el nivel de privilegio del segmento es 00. Por lo tanto, se encontrará en el área global, en la zona de PL = 0, como se recoge en el siguiente diagrama.

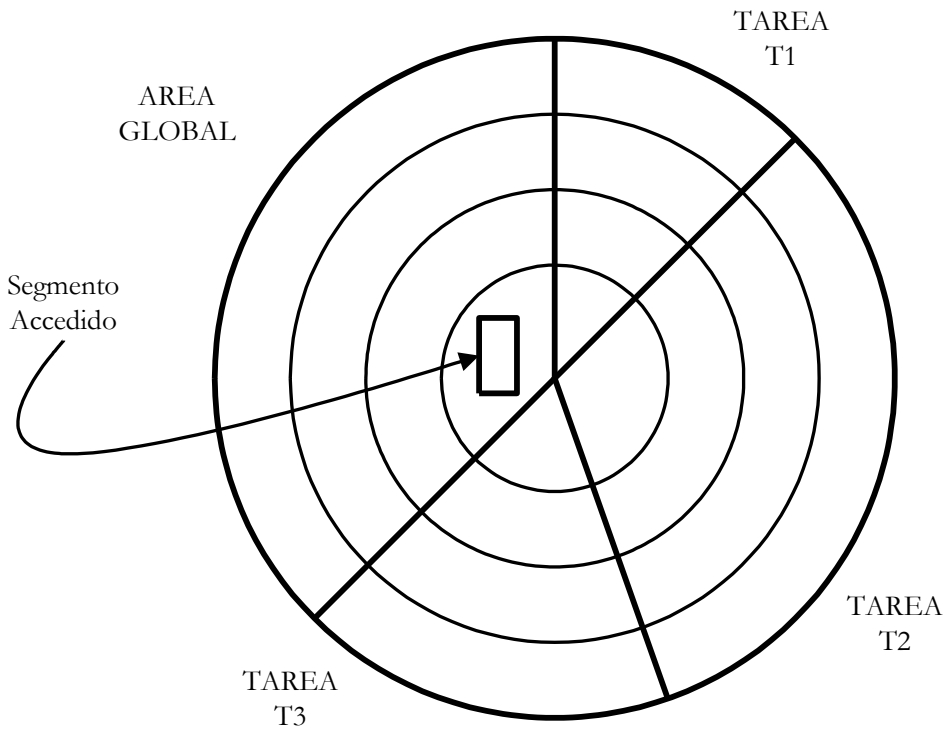



Figura 6 - Ubicación del segmento

**c) ¿Cuál es la dirección de la base del segmento?**
Teniendo en cuenta la figura 4, vemos claramente que la base de este segmento es:

00 40 00 07

**d) ¿Cuál es el tamaño del segmento?**

Si observamos el descriptor de segmento, vemos que su limite es 0 00 01. Sin embargo, al estar el bit G a 1 (activo) este límite no viene expresado en bits, sino en páginas de 4Kb. El tamaño del segmento es, por tanto, de 4Kb.

**e) ¿Está presente el segmento en memoria?**

El bit P está a 1 (activo), con lo que el segmento sí está presente en memoria.

**f) ¿De qué tipo de segmento se trata?**

Como el primer bit del campo 'tipo' de los atributos está a 1, se trata de un segmento de código, y los dos bits siguientes indican, respectivamente, que se trata de un segmento Conforming (o ajustable) y que puede leerse.

**g) ¿Cuál es la dirección física a la que se accede?**

Para obtener la dirección física, tomamos la base del segmento accedido, D0 00 92 A0, y le sumamos el desplazamiento que indica la dirección virtual, es decir:

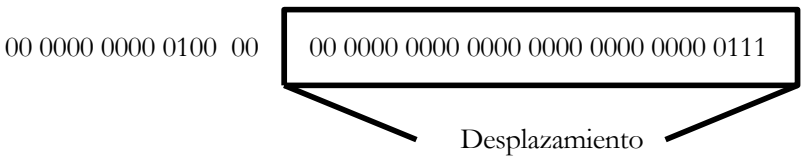



Figura 7 - Desplazamiento en una dirección virtual

Que en hexadecimal queda 00 00 00 07.

Por lo tanto, la dirección física accedida es 00 40 00 00 + 00 00 00 07 = 00 40 00 07

## 1.2. Ejercicio 2

**Se proporcionan a continuación los datos de cuatro segmentos de un entorno multitarea en el que conviven tres tareas (T1, T2 y T3, siendo T1 la activa), además de una dirección virtual para cada uno de ellos. Se pide calcular sus descriptores y construir, en base a ellos, las tablas de descriptores (GDT y LDT de la tarea en curso), expresando para todos los segmentos el límite en unidades de 4Kb, y dibujar asimismo la situación de memoria correspondiente.**

**a) Segmento A: Es un segmento de datos normal, asociado a software de 32 bits, cuyo nivel de privilegio es 2. Ocupa 64 Kb, y la última vez que estuvo en memoria, comenzaba en la dirección AA AA 00 00. Es un segmento escribible. Se corresponde con la dirección virtual**

00 0000 0000 0100 1000 0000 0000 0000 0000 0000 0000 0111

**b) Segmento B: Es un segmento de código normal de software de 32 bits, de máximo nivel de privilegio. Comienza en la dirección 01 00 80 00 y ocupa 64Kb. Es un segmento ajustable y leíble. Se corresponde con la dirección virtual**

00 0000 0000 0011 0000 0000 0000 0000 0000 0000 0000 0111

**c) Segmento C: Es el segmento de pila correspondiente al segmento C. Ocupa 32Kb, comenzando en la dirección en que acaba el segmento B. Se corresponde con la dirección virtual**

00 0000 0000 0101 0000 0000 0000 0000 0000 0000 0000 0111

**d) Segmento D: Es un segmento de código que contiene rutinas de sistema. Comienza en la dirección 02 00 00 00 y ocupa 128Kb. Es un segmento ajustable pero no puede ser leído. Se corresponde con la dirección virtual**

00 0000 0000 0010 0000 0000 0000 0000 0000 0000 0000 0111

## Resolución

Primeramente, conviene recordar la estructura de los descriptores de segmento (ver figura 4) y la colocación de los bits de atributo (figura 5), así como la división en campos de una dirección virtual (figuras 2 y 26). Una vez hecho esto, resolver este ejercicio es tan sencillo como cambiar los datos de esas figuras por los del enunciado.

a) Segmento A
**Es un segmento de datos normal, y es escribible:** Por ser un segmento de datos, los dos primeros bits del campo tipo (C/D# y ED) estarán a cero; por ser escribible, el tercero estará a 1. Por ser un segmento normal, el bit S estará a 1.

**Asociado a software de 32 bits:** Esto indica que el bit D/B estará a 1, ya que es un segmento nativo.

**Su nivel de privilegio es 2,** por lo que el campo DPL contendrá 10

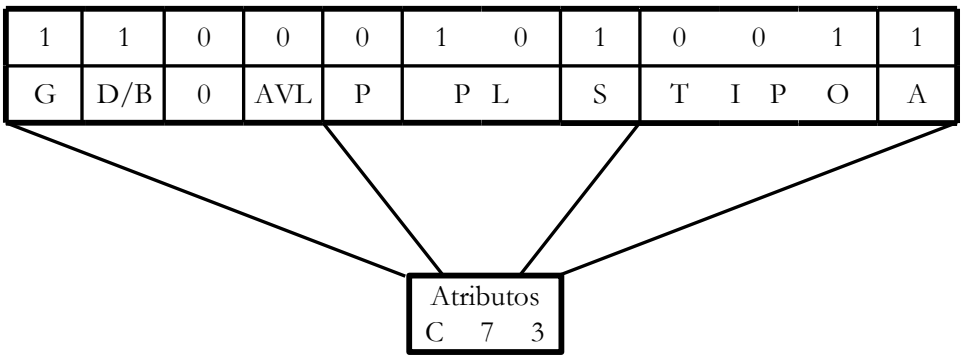
**Ocupa 64 Kb,** lo cual equivale a 16 unidades de 4Kb, por lo que el límite contendrá $00010_{)h}$

**La última vez que estuvo en memoria, comenzaba en la dirección AA AA 00 00,** así que esto será lo que contenga el campo base. Al no residir actualmente en memoria, el bit P estará a 0

**Se corresponde con la dirección virtual**
00 0000 0000 0100 1000 0000 0000 0000 0000 0000 0000 0111

Si separamos esta dirección virtual en sus campos, descubrimos que accede al tercer selector (selector número 2) de la GDT, por lo que este segmento estará ubicado en el área global.

Con estos datos, ya podemos construir establecer sus atributos y su descriptor (figura 8)




| | | | |
|---|---|---|---|
| AA | C0 | 73 | AA |
| 00 | 00 | 00 | 10 |

Figura 8 - Atributos y descriptor del segmento A

b) Segmento B

**Es un segmento de código normal, ajustable y leíble:** Los tres bits del campo tipo estarán a uno, el primero por ser un segmento de código, el segundo, por ser ajustable; y el tercero, por ser leíble. Por ser un segmento normal, el bit S estará a 1.

**Asociado a software de 32 bits** Esto indica que el bit D/B estará a 1, ya que es un segmento nativo.

**De máximo nivel de privilegio,** es decir, nivel 0. Por lo tanto, el campo DPL contendrá 00.

**Comienza en la dirección 01 00 80 00**, que será lo que contenga el campo base. Además, el bit P estará a uno.

**Ocupa 64Kb**, lo cual equivale a 16 unidades de 4Kb, con lo cual el campo límite contendrá $00010_{)h}$

**Se corresponde con la dirección virtual**

00 0000 0000 0011 0000 0000 0000 0000 0000 0000 0000 0111

Si separamos esta dirección virtual en sus campos, descubrimos que accede al segundo selector (selector 1) de la LDT, por lo que este segmento estará ubicado en el área local de la Tarea 1.

Con estos datos, ya podemos construir establecer sus bits de atributos y su descriptor (figura 9).

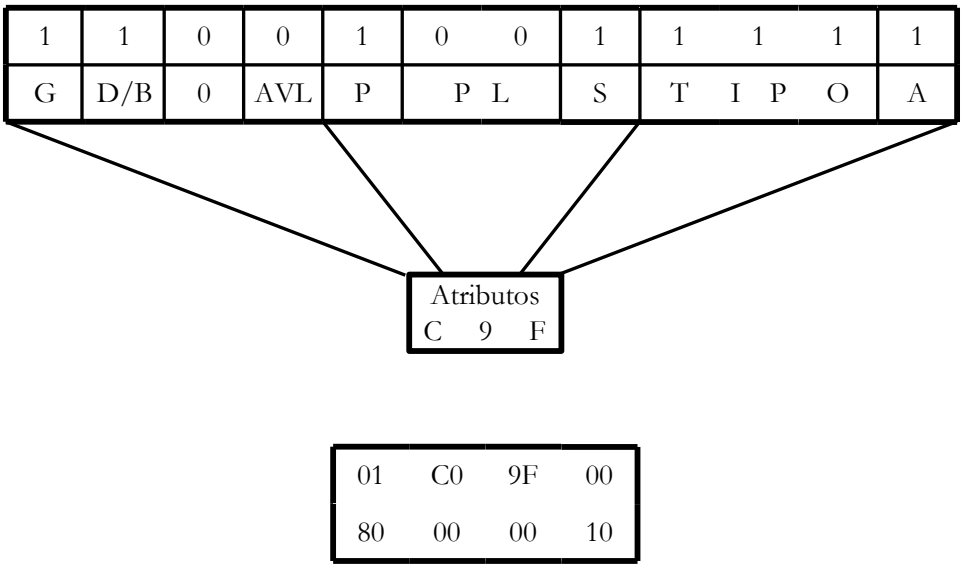



Figura 9 - Atributos y descriptor del segmento B

c) Segmento C

**Es el segmento de pila correspondiente al segmento C.** Dado que es un segmento de pila, su campo tipo tendrá el bit C/D# a 0, el bit ED a 1 y el bit W a 1, ya que los segmentos de pila crecen hacia abajo y pueden modificarse. Al estar asociado al segmento B, tendrá su mismo nivel de privilegio, es decir, 00.

**Ocupa 32Kb**, es decir, 8 unidades de 4Kb, por lo que su campo límite contendrá 00008

**Comienza en la dirección en que acaba el segmento B.** Para hallarla sumamos a la base del segmento B, 01 00 80 00, su tamaño, 00010, con lo que obtenemos 01 00 80 10, que será la dirección de comienzo del segmento C

**Se corresponde con la dirección virtual**

00 0000 0000 0101 0000 0000 0000 0000 0000 0000 0000 0111

Es decir, corresponde al tercer descriptor de la LDT, por lo que estará ubicado en el área local de la tarea T1.

Sus atributos y su descriptor quedan reflejados en la figura 10

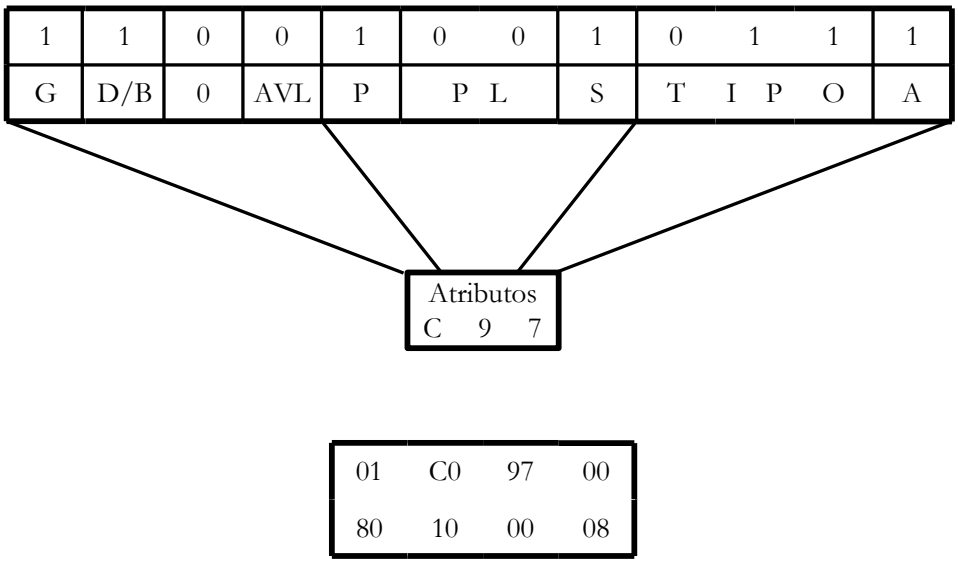



Figura 10 - Atributos y descriptor del segmento C

d) Segmento D

**Es un segmento de código que contiene rutinas de sistema, ajustable pero que no puede ser leído.** Al ser un segmento de rutinas del sistema, podemos deducir que su nivel de privilegio será el máximo (00). Además, al ser de código, su bit C/D# será uno; al ser ajustable, el bit Conforming también será uno, pero el bit de lectura estará a cero.

**Comienza en la dirección 02 00 00 00**, que será lo que contenga el campo base.

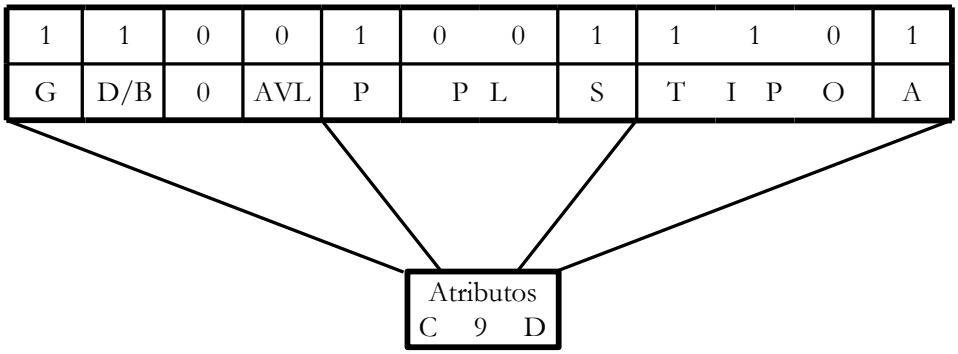
**Ocupa 128Kb**, es decir, 32 unidades de 4Kb, con lo que el campo límite contendrá $00020_{)h}$

**Se corresponde con la dirección virtual**

00 0000 0000 0010 0000 0000 0000 0000 0000 0000 0000 0111

Observamos que accede al segundo selector (selector número 1) de la GDT, por lo que el segmento estará ubicado en el área global.

En base a esto, construimos el descriptor reflejado en la figura 11




| | | | |
|---|---|---|---|
| 02 | C0 | 9D | 00 |
| 00 | 00 | 00 | 20 |

Figura 11 - Atributos y descriptor del segmento D

Una vez construidos los descriptores de los cuatro segmentos, podemos rellenar las tablas (GDT y LDT de la tarea T1), recordando incluir el descriptor nulo en la primera posición de la GDT. Así, las tablas quedan como indica la figura 12 (los guiones indican campos para los que no disponemos de datos)

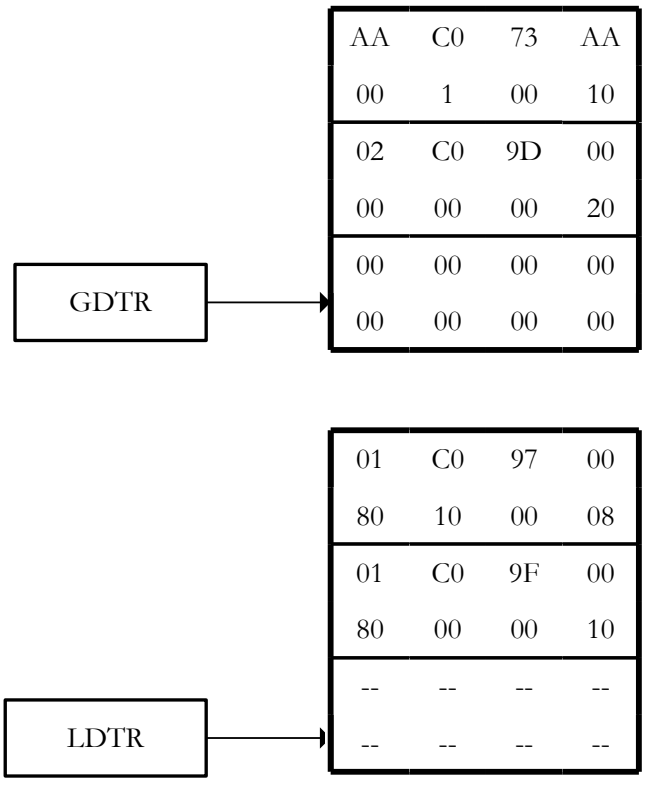



Figura 12 - GDT y LDT para la tarea T1

Por último, dibujamos la situación de memoria que reflejan estas tablas (figura 13)

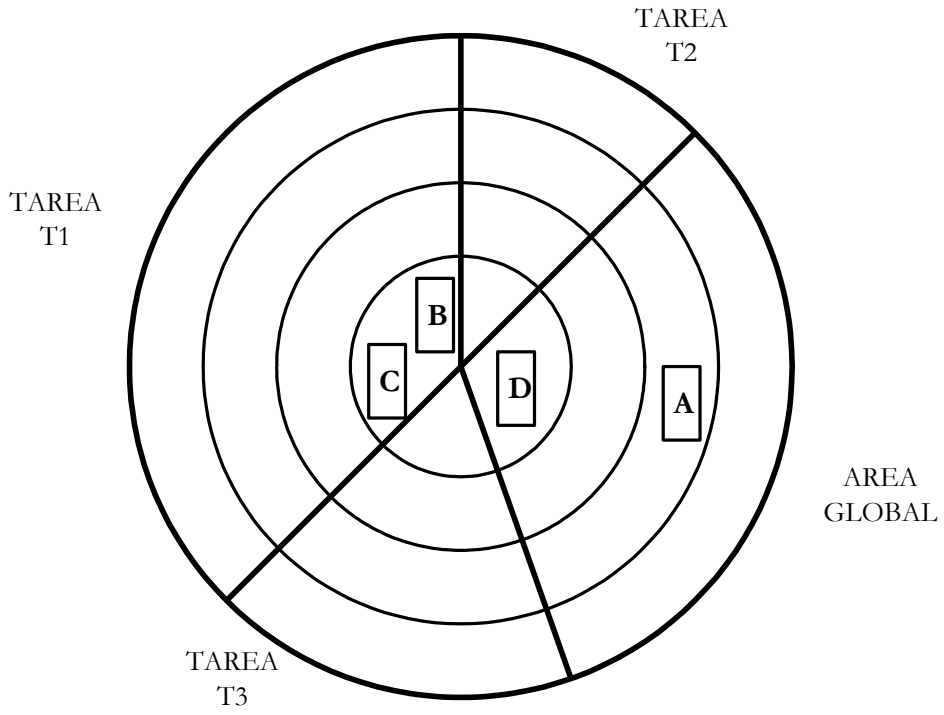



Figura 13 - Situación de memoria

## 1.3. Ejercicio 3

**Se trata ahora de realizar el proceso contrario: en lugar de disponer de una dirección virtual y hallar la dirección física a la que corresponde, partiremos de la dirección física accedida y será necesario deducir cuál fue la dirección virtual que provocó ese acceso, además de rellenar, en la medida de lo posible, el descriptor del segmento accedido. Para ello disponemos de los siguientes datos:**

- **Se accedió a la instrucción que ocupa la dirección FF 00 00 FF, que pertenece a una rutina del sistema operativo a la que tienen acceso todas las tareas.**
- **El segmento accedido puede ajustar su nivel de privilegio y no puede ser leído.**
- **La base del segmento es múltiplo de 64Kb (consideraremos que es el múltiplo más cercano), que es también el tamaño del mismo. Se trabaja con unidades de 4Kb para expresar los límites.**
- **Se accede al primer descriptor válido de la tabla que se utilice.**
- **No es un segmento de sistema, y está asociado a código de 32 bits**

### Resolución

Antes de resolver este ejercicio, sería conveniente recordar brevemente el funcionamiento de la segmentación (figura 14)

Partimos de la dirección accedida: FF 00 00 FF Sabemos que la dirección de la base del segmento es múltiplo de 64Kb = $10000_{)h}$, por lo que debe acabar en cuatro ceros hexadecimales Por lo tanto, consideraremos que la base es FF 00 00 00, y por tanto el desplazamiento será, en hexadecimal, 00 00 00 FF.

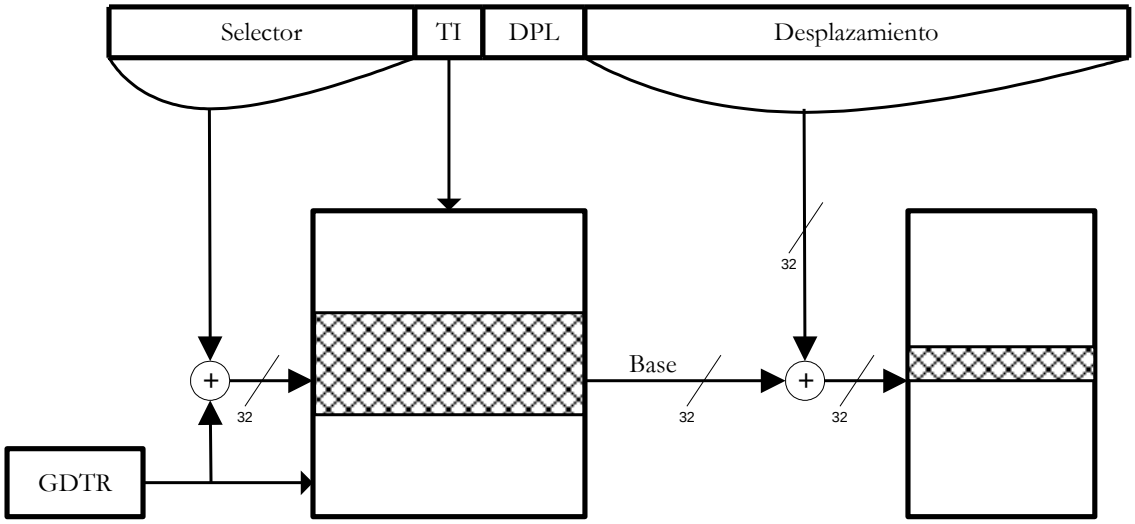



Figura 14 - Funcionamiento de la segmentación

Dado que la rutina de la que forma parte esta instrucción es accesible por todas las tareas, deducimos que está en el área global de memoria, por lo que su descriptor estará en la GDT, y el bit TI de la dirección virtual será 0. Además, por el momento consideraremos que tan sólo un segmento de código de nivel 0 puede acceder a otro de ese nivel (más adelante veremos que existen recursos que permiten saltarse esta regla de protección), por lo que el RPL que aparece en la dirección virtual será 00

El primer descriptor válido de la GDT es el segundo (el número 1), con lo que el selector que le corresponde es el $00\ 0000\ 0000\ 0001_{)2}$

Podemos ahora, por tanto, rellenar el esquema con los datos que hemos obtenido:

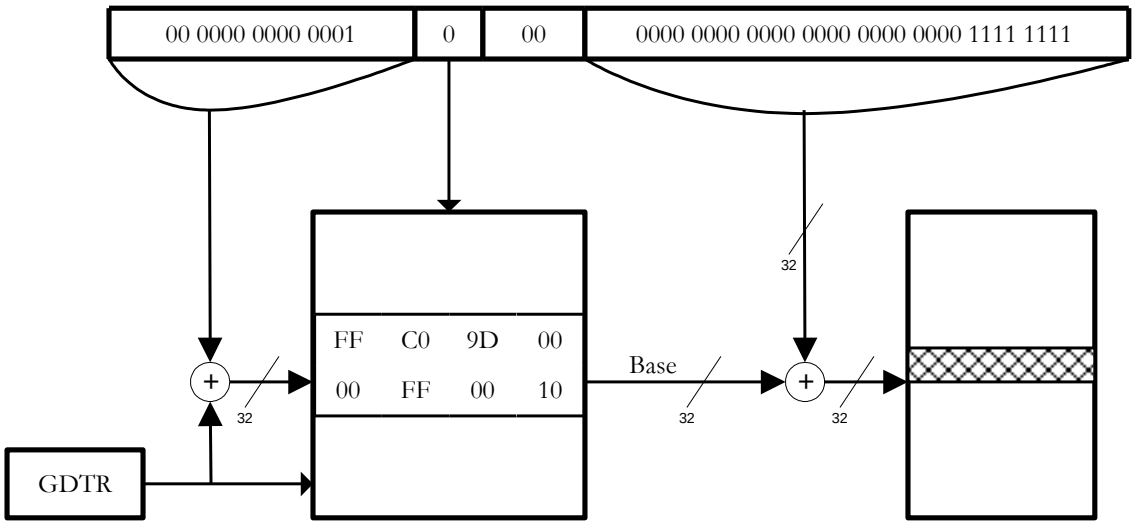



Figura 15 - Dirección virtual y descriptor correspondiente

## 1.4. Ejercicio 4

En un entorno multitarea hay tres tareas en memoria (T1, T2 y T3), estando activa la tarea T2. En un instante dado, genera la dirección virtual (en binario):

00 0000 0000 0010 0000 0000 0000 0000 0000 0000 0000 0111

En la primera posición útil de las tablas de segmentos hay información relativa a los dos segmentos que se describen a continuación, pudiendo existir además descriptores relativos a otros.

Segmento 1: Es un segmento de código normal de software de 32 bits, correspondiente a software de usuario. Comienza en la dirección AA AA 00 00 y ocupa 64Kb. Es un segmento no ajustable y no leíble, exclusivo de la tarea T2.

Segmento 2: Es un segmento de código que contiene rutinas de sistema. ocupa 128Kb y acaba donde comienza el segmento 1. Es un segmento ajustable pero no puede ser leído. Puede ser accedido por todas las tareas.

a) ¿Al descriptor de  qué segmento se accede?
b) ¿Dónde se encuentra el segmento accedido? Realizar un diagrama de la memoria que muestre claramente dónde se encuentran ambos segmentos y a cuál de ellos se accede.
c) ¿Cuál es el tamaño del segmento 2?
d) ¿Cuál es la dirección física a la que se accede? Realizar un diagrama que explique el funcionamiento de la segmentación en este caso concreto.

# 2. TEMA 9: PAGINACIÓN

## 2.1. Ejercicio 1

**Supongamos la situación del Directorio y las Tablas de Páginas que muestra la figura 26.16. Teniendo en cuenta los datos que proporciona el diagrama, responder a las preguntas siguientes:**

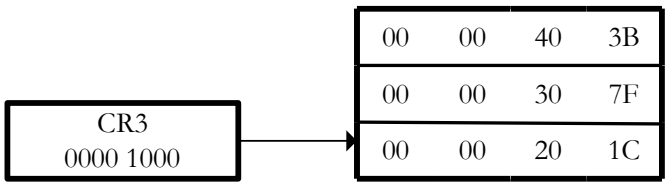



Tabla de páginas 0

| | | | |
|---|---|---|---|
| 00 | 00 | 60 | 7C |
| 00 | 00 | 50 | 1C |

Tabla de páginas 1

| | | | |
|---|---|---|---|
| 00 | 00 | 80 | 1C |
| 00 | 00 | 70 | 7F |

Tabla de páginas 2

| | | | |
|---|---|---|---|
| 00 | 00 | A0 | 7C |
| 00 | 00 | 90 | 1F |

Figura 16 - Directorio y Tablas de Páginas

**a) ¿Cuáles de las páginas están presentes en memoria? ¿Cuáles han sido accedidas? ¿Cuáles han sido modificadas?**
**b) ¿Cuál es el nivel de privilegio de las páginas de la tabla de paginas 0?**
**c) ¿Cuál es la dirección de comienzo de cada una de las tablas de páginas?**

**d) ¿Cuáles de las páginas son cacheables?**
**e) ¿Qué tamaño tiene cada una de las páginas?**

## Resolución

Antes de resolver las preguntas, conviene recordar cuál es la estructura de las entradas del directorio y de las tablas de páginas ( figura 17).

| Direccion (20 bits) | 0000 | SIZ/0 | D | A | PCD | PWT | U/S | R/W | P |
|---|---|---|---|---|---|---|---|---|---|

Figura 17 - Estructura de una entrada de directorio

**a) ¿Cuáles de las páginas están presentes en memoria? ¿Cuáles han sido accedidas? ¿Cuáles han sido modificadas?.**

Para responder a estas preguntas, analizaremos los atributos de cada entrada, que aparecen reflejados en la figura 18

| Direccion (20 bits) | 0000 | SIZ/0 | D | A | PCD | PWT | U/S | R/W | P |
|---|---|---|---|---|---|---|---|---|---|
| Entrada 0 del Directorio | | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| Entrada 1 del Directorio | | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Entrada 2 del Directorio | | 0 | 0 | 1 | 1 | 1 | 0 | 1 | 1 |
| Entrada 0 de Tabla 0 | | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| Entrada 1 de Tabla 0 | | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |
| Entrada 0 de Tabla 1 | | 0 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| Entrada 1 de Tabla 1 | | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 1 |
| Entrada 0 de Tabla 2 | | 0 | 0 | 0 | 1 | 1 | 1 | 1 | 1 |
| Entrada 1 de Tabla 2 | | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 0 |

Figura 18 - Atributos en las entradas del Directorio y las Tablas de Páginas

Están presentes, por tanto, las páginas referenciadas por la entrada 1 de la tabla de páginas 1 y la entrada 0 de la tabla 2, así como las entradas 1 y 2 del directorio, es decir, las que contienen las tablas de páginas 1 y 2.

Han sido accedidas las páginas referenciadas por la entrada 1 de las Tablas 0, 1 y 2, así como las entradas 1 y 2 del directorio. Todas esas páginas, salvo la entrada 2 del Directorio, han sido, además, modificadas.

**b) ¿Cuál es el nivel de privilegio de las páginas de la tabla de paginas 0?**

Como vemos en la figura 18, para ambas páginas el nivel de privilegio es uno, es decir, son páginas de usuario.

**c) ¿Cuál es la dirección de comienzo de cada una de las tablas de páginas?**

Para responder a esta pregunta es necesario observar la parte de la entrada que no hemos contemplado en la figura 18, que nos da los 20 bits más significativos de la dirección. Dado que todas las páginas son de 4Kb, al estar el bit SIZ de CR3 a cero, los 12 bits restantes serán ceros. Por lo tanto, la dirección de comienzo de la tabla 0 será la 00 00 20 00, la de la tabla 1 será 00 00 30 00 y la de la tabla 2 será 00 00 40 00. Se observa que están colocadas consecutivamente en memoria

**d) ¿Cuáles de las páginas son cacheables?**

Observando el bit PCD de los atributos de las entradas (figura 18) , podemos ver que son cacheables todas las páginas.

**e) ¿Qué tamaño tiene cada una de las páginas?**

Como ya se ha dicho, dado que el bit SIZ de CR3 está a 0, todas las páginas son de 4Kb.

## 2.2. Ejercicio 2

**Retomemos el ejercicio del apartado 1.1. Supongamos que, con la misma situación para las tablas de descriptores (figura 1), se activa la paginación. En base a la información relativa al Directorio de Tablas de Páginas y las Tablas de Páginas que se proporciona en la figura 16,responder a las siguientes preguntas.**

**a) ¿ Cuál es la dirección lineal que se obtiene a partir de la dirección virtual?**
**b) ¿ A qué entrada del Directorio se accede?**
**c) ¿ A qué tabla de páginas se accede?**
**d) ¿ Cuál es la dirección física que se corresponde con la dirección virtual inicial?**

### Resolución

**a) ¿ Cuál es la dirección lineal que se obtiene a partir de la dirección virtual?**

Recordemos que, cuando se activa la paginación, el Pentium trabaja con Segmentación Paginada. Por lo tanto, el primer paso de la traducción de la dirección virtual es exactamente el que se hizo en el ejercicio 1.1. La diferencia es que, utilizando sólo segmentación, el valor obtenido en dicha traducción (00 40 00 07) es la dirección física a la que se accede. En este caso, sin embargo, es la dirección lineal, que se utiliza para acceder al directorio de páginas y desde él a la tabla que corresponda, hasta obtener la dirección física.

**b) ¿ A qué entrada del Directorio se accede?**

Para saberlo, basta descomponer la dirección lineal en sus campos, como indica la figura 19. Vemos que se accede a la segunda entrada (entrada numero 1) del directorio, y, por tanto, a la Tabla de Páginas 1.

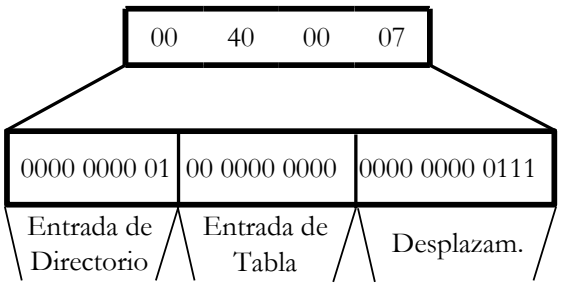



Figura 19 - División de una dirección lineal

**c) ¿ A qué tabla de páginas se accede?**

A la vista de la figura anterior, vemos que se accede a la primera entrada (numero 0) de la Tabla de Páginas 1.

**d) ¿ Cuál es la dirección física que se corresponde con la dirección virtual inicial?**

Para obtener la dirección física, habría que tomar la dirección base que indica la entrada a la que se ha accedido, 00007, y añadirle los doce ceros que no se almacenan en las tablas al ser todas las páginas múltiplos de 4Kb. Obtenemos, por lo tanto, 00 00 70 00. Si a esta dirección base le sumamos el desplazamiento, $007_{)h}$, obtenemos las dirección física a la que se accede, que en este caso es 00 00 70 07. También puede obtenerse directamente concatenando el desplazamiento a los bits de la dirección que proporciona la tabla.

## 2.3. Ejercicio 3

**Se trata ahora de realizar el proceso contrario: conociendo la dirección a la que se accede, la FF 00 00 FF, hallar la dirección lineal de la que se partió. Rellenar también, en la medida de lo posible, las entradas del Directorio y de las Tablas de Páginas por las que se pase. Para ello sabemos que:**

- **Se está trabajando con páginas de 4Kb**
- **El desplazamiento dentro de la Tabla de Páginas y del Directorio para acceder a la entrada que corresponde es de 256 bites.**

### Resolución

Para completar la dirección virtual necesitamos conocer el número de entrada del Directorio y de la Tabla de Paginas que se utilizan, además del desplazamiento.

Comenzaremos por éste último. Como las páginas ocupan 4Kb, sus direcciones de comienzo serán múltiplos de ese valor, es decir, acabarán en doce ceros binarios. Por lo tanto, deducimos que la dirección de la base de la página a la que se accede es FF 00 00 00, y el desplazamiento será, por tanto, la diferencia entre la dirección física y la base, es decir,

$$\text{FF 00 00 FF} - \text{FF 00 00 00} = \text{00 00 00 FF}$$

Dado que solo tomamos los doce bits menos significativos, el valor a colocar en el campo desplazamiento de la dirección lineal será 0FF

Para hallar el número de entrada de la Tabla de Páginas y del Directorio conocemos el desplazamiento en bits, 256. Dado que cada entrada ocupa 32 bits, en ambos casos será la novena entrada, es decir, la número 8. Por lo tanto, en ambos campos de la dirección lineal colocaremos el valor $00\ 0000\ 1000_{)b}$. Así, la dirección lineal completa queda:

$$0000\ 0010\ 0000\ 0000\ 1000\ 0000\ 1111\ 1111_{)2} = 02\ 00\ 80\ FF_{)h}$$

En cuanto a la entrada de la Tabla de Páginas, con los datos de los que disponemos podemos rellenar el campo base, pero no los atributos (figura 20). De la entrada del Directorio, sin embargo, no podemos dar ningún valor.

| | | | |
|---|---|---|---|
| FF | 00 | 0 - | - - |
| - - | - - | - - | - - |
| - - | - - | - - | - - |
| - - | - - | - - | - - |
| - - | - - | - - | - - |
| - - | - - | - - | - - |
| - - | - - | - - | - - |
| - - | - - | - - | - - |
| - - | - - | - - | - - |

Figura 20 - Tabla de páginas a la que se accede

## 2.4. Ejercicio 4

Suponiendo la misma situación de memoria que en el ejercicio 2.1, responder a las siguientes preguntas:

a) ¿Qué habría que modificar en la GDT para que se accediera a la primera página de la tercera tabla de páginas?

b) Realizar un diagrama que muestre el funcionamiento tanto de la segmentación como de la paginación en este caso, rellenando todos los campos posibles

c) ¿Qué dirección lineal se corresponde con ese acceso?

# 3. TEMA 11: MECANISMO DE PROTECCIÓN

## 3.1. Ejercicio 1

**Dado el esquema de memoria de la figura 21, en el que C denota un segmento de código, D de datos y P de pila, determinar:**

**a) A qué segmentos de código se puede acceder desde C0**
**b) A qué segmentos de pila se puede acceder desde C0**
**c) A qué segmentos de datos se puede acceder desde C0**
**d) A qué segmentos de código se pude acceder desde C1**
**e) A qué segmentos de datos se puede acceder desde C1**

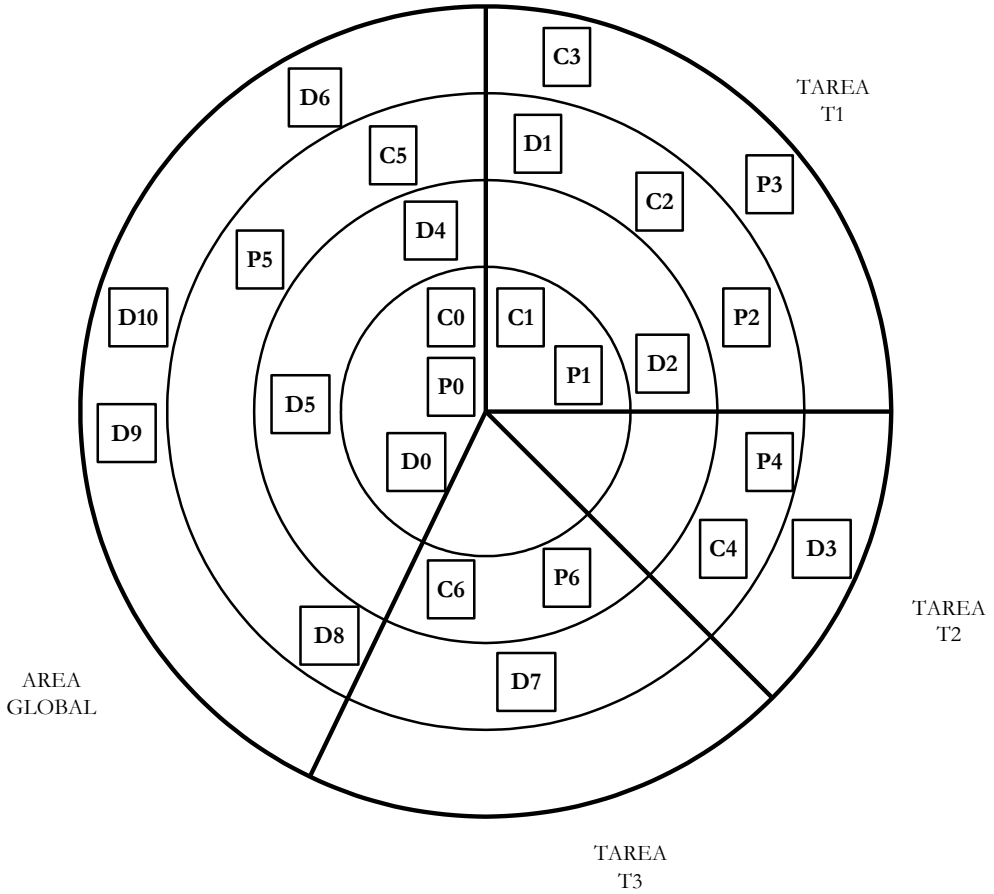



Figura 21 - Situación de memoria en un entorno multitarea

### Resolución

**a) A qué segmentos de código se puede acceder desde C0**

Observamos que C0 se encuentra en el nivel de privilegio 0 del área local. Podría acceder a otros segmentos de código del área global y del nivel 0. Dado que no hay ninguno más, no puede acceder a ningún segmento de código. (Conviene recordar que desde el área global no puede accederse al área local de una tarea)

**b) A qué segmentos de pila se puede acceder desde C0**

Los segmentos de pila siguen la misma regla de acceso que los de código. Así, el segmento C0 puede acceder al segmento P0, que está en su misma área de memoria y mismo nivel de privilegio.

**c) A qué segmentos de datos se puede acceder desde C0**

Un segmento de código puede acceder a segmentos de datos de su misma área o del área global de nivel de privilegio igual o menor al suyo propio. En este caso, el segmento C0 podrá acceder a todos los segmentos de datos del área global, es decir: D0, D4, D5, D8, D6, D9 y D10.

**d) A qué segmentos de código se pude acceder desde C1**

C1 puede acceder al segmento C0, ya que se encuentra en su mismo nivel de privilegio y en el área global.

**e) A qué segmentos de datos se puede acceder desde C1**

C1 puede acceder a todos los segmentos de datos tanto del área local como de la suya propia, es decir, D0, D4, D5, D8, D6, D9 y D10 en el área global, y D1 y D2 en su área local.

## 3.2. Ejercicio 2

**Dado el esquema de memoria de la figura 20, determinar desde qué segmentos se puede acceder a:**

**a) C1, estando activa la tarea T1**
**b) D4, estando activa la tarea T2**
**c) C5, estando activa la tarea T3**
**d) P4, estando activa la tarea T2**
**e) D0, estando activa la tarea T1**
**f) D0, estando activa la tarea T2**

### Resolución

**a) C1, estando activa la tarea T1**

Para poder acceder a T1, dado que se encuentra en el nivel de privilegio 0 de su área local, debería haber otro segmento de código en la misma área y nivel. Por lo tanto, no hay ningún segmento en la figura que pueda acceder a C1

**b) D4, estando activa la tarea T2**

D4 puede ser referenciado por el segmento C0 del área global. Ningún segmento de código de la tarea T2 puede acceder a él.

**c) C5, estando activa la tarea T3**

Dado que no hay ningún segmento de código al mismo nivel ni en el área global ni en la local a la tarea T3, ninguno de los segmentos de la figura puede acceder a C5 en esta situación.

**d) P4, estando activa la tarea T2**
P4 sólo puede ser accedido desde C4, ya que está en el área local a T2.

**e) D0, estando activa la tarea T1**
Puede ser accedido desde C1, que está en el área local de T1 y en el mismo nivel de privilegio.

**f) D0, estando activa la tarea T2**
No puede ser accedido desde ningún segmento, ni del área global ni de la tarea T2.

## 3.3. Ejercicio 3

**¿Dónde habría que colocar un segmento de código para que pudiera acceder simultáneamente a los siguientes conjuntos de segmentos? Puede suceder que haya varias ubicaciones diferentes para el mismo segmento, o que no exista ninguna.**

**a) D10, D5 y D7**
**b) D10, D5 y P6**
**c) P2 y P4**
**d) C0 y C6**

### Resolución

**a) D10, D5 y D7**
D10 y D5 están en el área global, y D7 en la de la Tarea T3, por lo que el segmento pedido debe encontrarse en el área de T3. Además, dado que D5 es el segmento de mayor nivel de privilegio (1), deberá estar bien en el nivel 1 o en el 0 de la tarea T3.

**b) D10, D5 y P6**
Al igual que antes, debe estar en el área de memoria reservada para T3 para poder acceder a P6. Además, dado que P6 es un segmento de pila, obliga a que esté situado en su mismo nivel de privilegio, es decir, en el nivel 1 de la tarea T3

**c) P2 y P4**
Dado que P2 y P4 pertenecen a tareas distintas, es imposible que un mismo segmento pueda acceder a ambas.

**d) C0 y C6**
C0 está en el área global y C6 en la de la tarea T3, por lo que, de existir, el segmento debería esta en esta última zona de memoria. Sin embargo, C0 está en el nivel de privilegio 0 y C6 en el 1, y dado que los segmentos de código sólo pueden ser accedidos por otros de su mismo nivel de privilegio, es imposible que un mismo segmento pueda acceder a ambos.

## 3.4. Ejercicio 4

Dada la situación de memoria de la figura 21, determinar

a) A qué segmentos de datos puede accederse desde C6
b) A qué segmentos puede accederse desde D0
c) Desde qué segmentos puede accederse a P5
d) ¿Hay algún segmento que pueda acceder a todos los demás presentes en memoria?

# 4. TEMAS 12 y 13: PUERTAS DE LLAMADA Y DE TAREA

## 4.1. Ejercicio 1

**Dado el esquema de memoria de la figura 22, responder a las siguientes preguntas**

**a) ¿Qué segmentos pueden usar DG2?**
**b) ¿Desde qué segmentos se puede acceder a CG1?**
**c) ¿Qué segmentos de código pueden utilizar la PLLG3?**
**d) ¿Puede acceder el segmento C10 al CG1?**
**e) ¿Desde qué segmentos de la tarea T2 se puede utilizar la PLL2?**

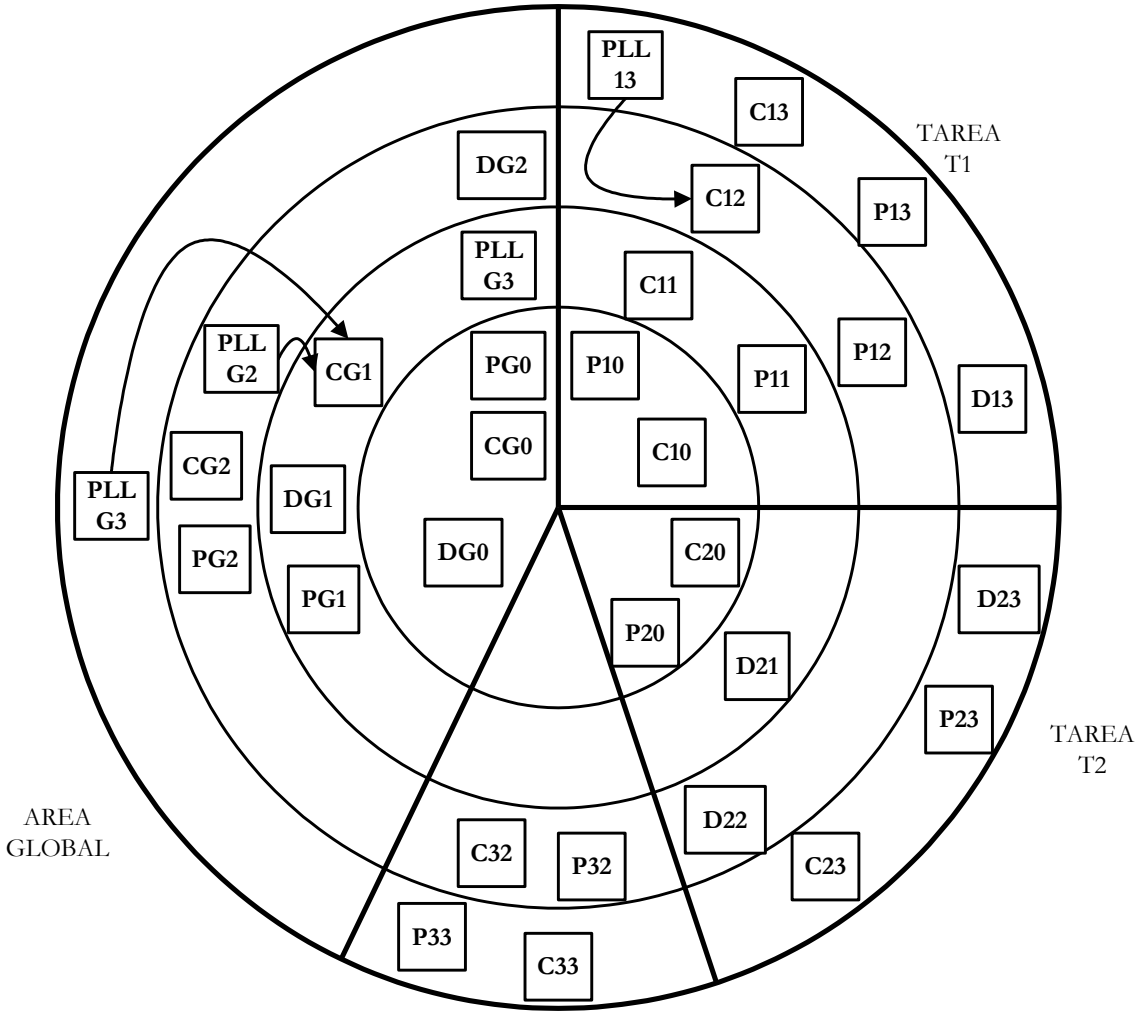



Figura 22 - Situación de memoria

### Resolución

**a) ¿Qué segmentos pueden usar DG2?**

Dado que es un segmento de datos, pueden acceder a él todos aquellos segmentos cuyo nivel de privilegio sea igual o superior a 2, es decir:

- Del área global, CG2, CG1, CG0.
- De la tarea T1, C12, C11, C10.
- De la tarea T2, C20
- De la tarea T3, C32.

**b) ¿Desde que segmentos se puede acceder a CG1?**

Dado que es un segmento de código, pueden acceder a él aquellos que se encuentren en su mismo nivel de privilegio. Dado que PLLG3 y PLLG2 proporcionan acceso a CG1, también habrá que tener en cuenta aquellos segmentos que puedan llamar a las PLLs y cuyo nivel de privilegio se vea aumentado al acceder a CG1.Teniendo esto en cuenta, la lista de segmentos que pueden acceder a CG1 es.:

- Directamente: C11
- A través de PLLG3: CG2, C32, C33, C23, C12  C13
- A través de PLLG2: CG2, C12,

En conclusión: C11, CG2, C32, C33, C23, C12 y C13

**c) ¿Qué segmentos de código pueden utilizar la PLLG3?**

Como ya hemos dicho en la pregunta anterior: CG2, C32, C33, C23, C12  C13

**d) ¿Puede acceder el segmento C10 al CG1?**

No puede. Directamente el acceso no está permitido, ya que el nivel de privilegio de CG1 es inferior al de C10. C10 puede llamar a PLLG3 y PLLG2., que le permitiría acceder a CG1; sin embargo, dado que con dicho acceso su nivel de privilegio no resulta aumentado, tampoco es posible.

**e) ¿Desde qué segmentos de la tarea T2 se puede utilizar la PLLG2?**

Desde ninguno, ya que no hay ningún segmento que tenga el suficiente nivel de privilegio para acceder a ella y que a su vez resulte aumentado por el acceso al segmento CG1 desde la puerta.

## 4.2. Ejercicio 2

**Dados los descriptores de la figura 23, indicar, para cada uno de ellos, si corresponde a una puerta de llamada o de tarea, y el valor de cada uno de sus campos.**

Puerta A

| | | | |
|---|---|---|---|
| 00 | 80 | EC | 1F |
| 00 | 01 | 00 | 07 |

Puerta B

| | | | |
|---|---|---|---|
| 00 | 80 | E9 | 1F |
| 00 | 01 | 00 | 07 |

Puerta C

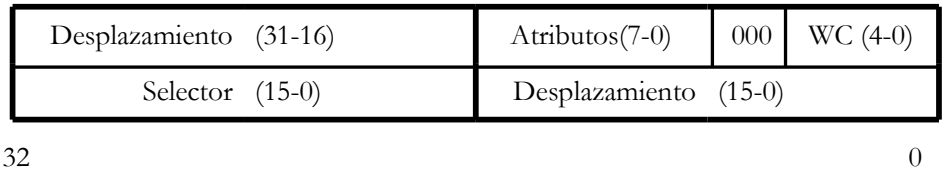
| | | | |
|---|---|---|---|
| 00 | 80 | EB | 1F |
| 00 | 01 | 00 | 07 |

Figura 23 - Descriptores de puertas

## Resolución

Antes de resolver este ejercicio, convendría recordar cuál es la estructura de una PLL y de una PT (figura 24), así como que el campo tipo tiene el valor $1100_{)b}$ para las puertas de llamada y $010B1_{)b}$ para las puertas de tarea. Una vez hecho esto, pasamos a analizar cada descriptor.

Puerta de Llamada




Puerta de Tarea

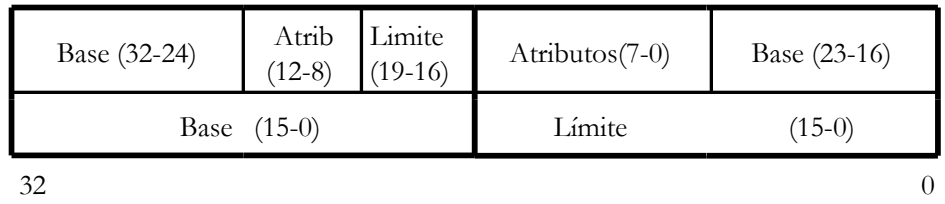



Figura 24 - Estructura de las puertas

- Puerta A

Si observamos los bits 0 a 7 del campo Atributos, observamos que es $EC_{)h} = 1110\ 1100_{)2}$, y dado que el tipo lo indican los cuatro bits de menos peso, se trata de una puerta de llamada. Por lo tanto, los demás campos del descriptor son:

- Selector: 00 01
- Desplazamiento: 00 80 00 07
- Word Counter (WC): es $11111_{)b}$, es decir, 31
- Atributos: Los atributos de un descriptor de puerta de tarea son el bit P, dos bits para el nivel de privilegio DPL, otro para indicar si es un segmento de sistema y el tipo. Según esto, se trata de una puerta de llamada de sistema que se encuentra en memoria cuyo nivel de privilegio es 3.

- Puerta B

Si observamos los bits 0 a 7 del campo Atributos, observamos que $E9_{)h} = 1110\ 1001_{)2}$, y dado que el tipo lo indican los cuatro bits de menos peso, se trata de una puerta de Tarea.. Por lo tanto, los demás campos del descriptor son:

- Base: 00 1F 00 01
- Límite: 00007
- Atributos: Son los mismos que en un descriptor normal. Observamos que el bit de granularidad está activo, por l oque el límite está expresado en unidades de 4Kb. El bit D/B está también a uno, indicando que se trata de un segmento asociado a código de 32 bits. Podemos decir además que está presente en memoria y que su nivel de privilegio es 3.
- Bit Busy: El cuarto bi del campo tipo indica si la tarea está o no ocupada. En este caso, está libre.

- Puerta C

Si comparamos este descriptor con el anterior, vemos que sólo difiere en un bit, que es precisamente el bit Busy. Todo lo dicho en el apartado anterior es válido para esta tarea, que sólo se diferencia en que está ocupada.

## 4.3. Ejercicio 3

Dado el esquema de memoria de la figura 22, responder a las siguientes preguntas:
a) ¿Qué segmentos deberían ser "ajustables" para evitar que se produzca el escenario del Caballo de Troya?
b) ¿Dónde habría que colocar una PLL para que todos los segmentos de código presentes en memoria pudieran acceder a CG0?
c) Explicar razonadamente por qué no tiene sentido colocar una PLL en el máximo nivel de privilegio

## 4.4. Ejercicio 4

Dado el esquema de memoria de la figura 25, responder a las siguientes preguntas:
a) ¿Existe algún error en el gráfico? En caso afirmativo, razonarlo.
b)¿ Desde qué segmentos de código se puede realizar una conmutación a la tarea T1?
c) ¿ Qué segmentos deberían hacerse "ajustables" para evitar que se produzca el escenario del caballo de Troya?
d) ¿Desde qué segmentos se puede acceder a C11?
e) ¿Qué segmentos pueden acceder a D12?
f) ¿Qué segmentos de la tarea T2 pueden acceder a C11?

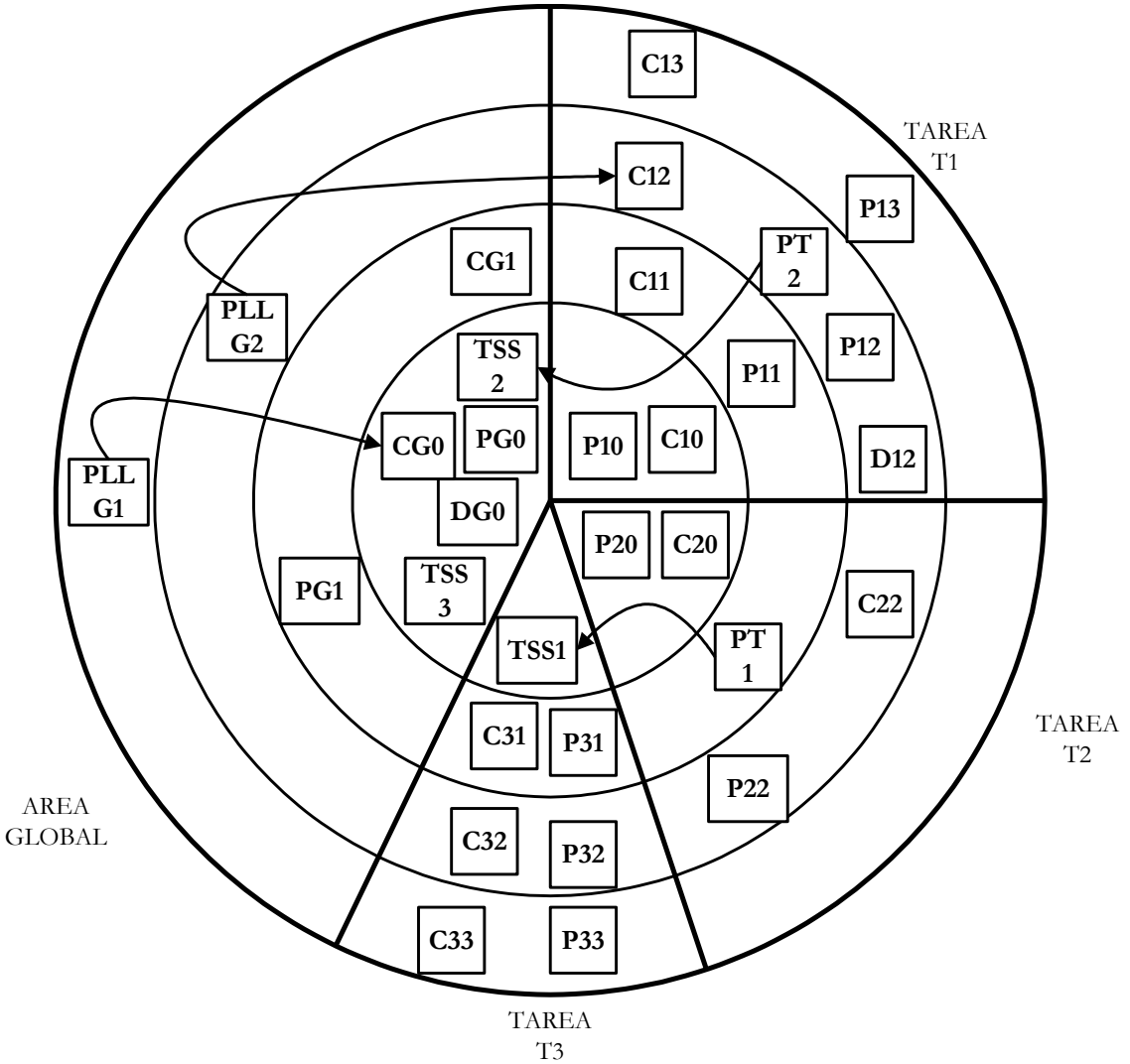



Figura 25 - Situación de memoria

# 5. TEMA 16: CICLOS DE BUS

## 5.1. Ejercicio 1

**Dibujar y explicar el diagrama de un ciclo de bus simple de lectura de memoria no cacheable, con un solo estado T2, representando también la línea de comprobación de paridad.**

### Resolución

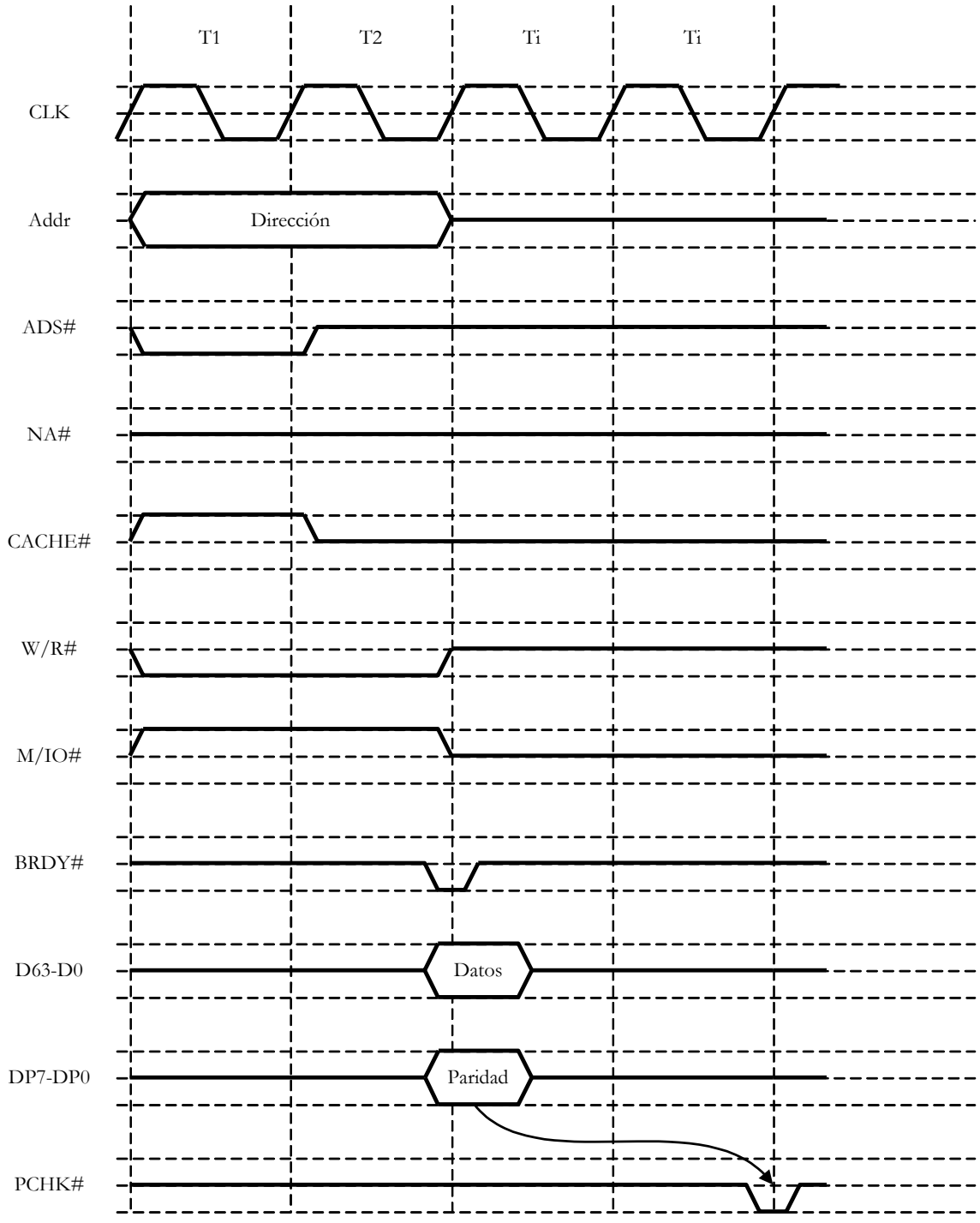



Figura 26- Ciclo de bus simple de lectura

La figura 26 muestra las líneas de control que definen el ciclo de bus. En el momento en que se activa ADS#, aparece en el bus de direcciones una dirección válida, se marca el ciclo como no cacheable poniendo a uno la línea CACHE#, se indica que es una lectura y que el acceso se realizará a memoria. Una vez completado el ciclo, la memoria activa la señal BRDY# para indicar que en el bus de datos ya están disponibles el dato pedido y su paridad. Dos ciclos de reloj después, el procesador activará PCHK# para indicar que ha realizado la comprobación de paridad.

## 5.2. Ejercicio 2

**Dibujar y explicar el diagrama de un ciclo de bus a ráfagas de escritura, representando también la línea de comprobación de paridad**

### Resolución

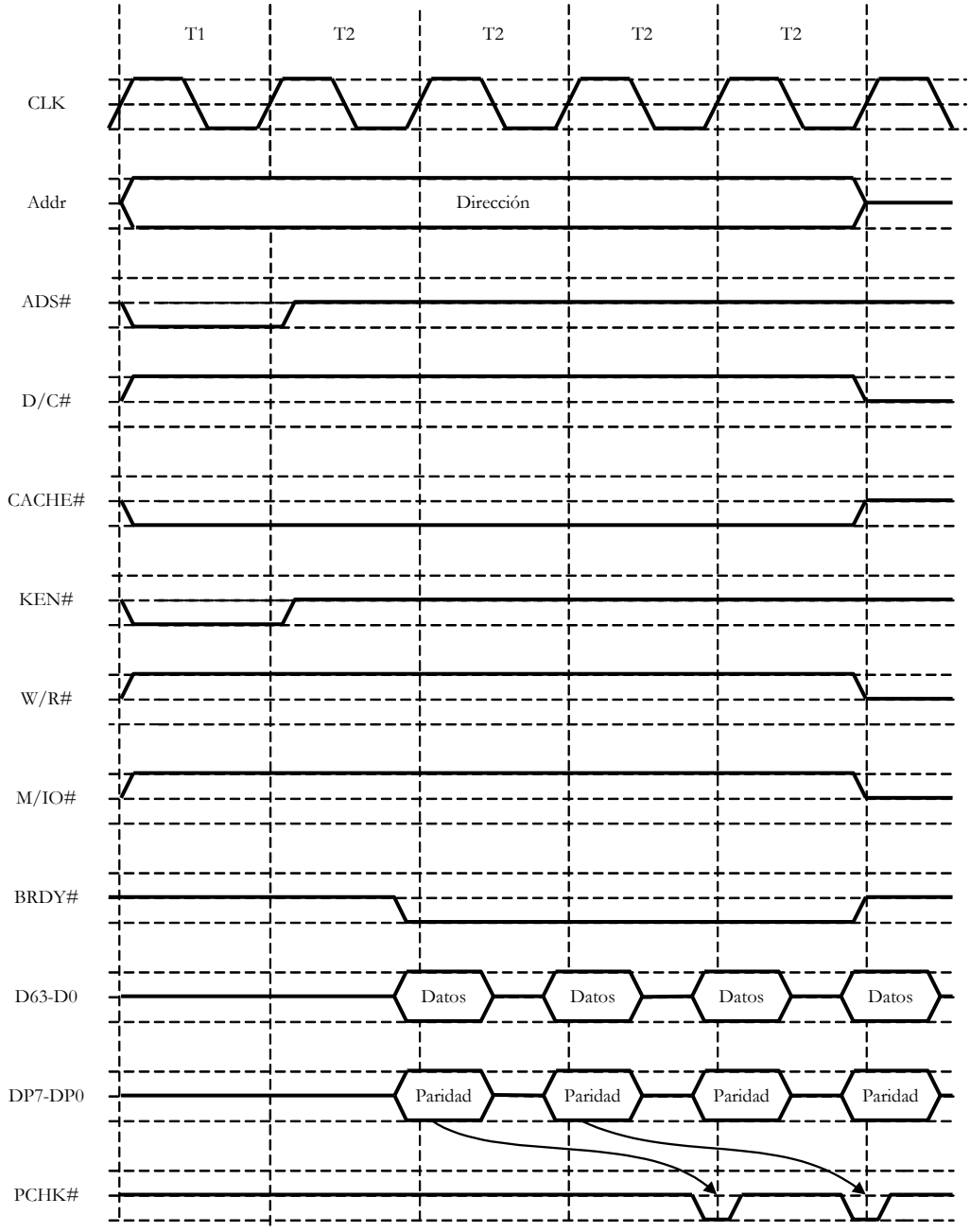



Figura 27 - Ciclo de bus a ráfagas

La figura 27 muestra las líneas de control que definen el ciclo de bus. A diferencia de un ciclo normal, uno a ráfagas consta de cinco estados, un T1 y cuatro T2 a continuación. En cada uno de ellos se genera un conjunto de datos con su paridad asociada, y la señal PCHK# se va generando dos ciclos después de ser recogido cada dato.

Como se trata de un ciclo de caché, tanto la señal CACHE# como KEN# deben estar activas. Recuérdese que los intercambios con caché siempre son con memoria, nunca con E/S, por lo que M/IO# debe estar a uno.

## 5.3. Ejercicio 3

Dibujar y explicar el diagrama de un ciclo de bus simple de escritura de memoria no cacheable, con tres estados T2, representando también la línea de comprobación de paridad.

## 5.4. Ejercicio 4

Dibujar y explicar el diagrama de un ciclo de bus a ráfagas de escritura, representando también la línea de comprobación de paridad.