Datos que deben extraerse del enunciado (sí o sí) para poder realizar los ejercicios de asignación no contigua

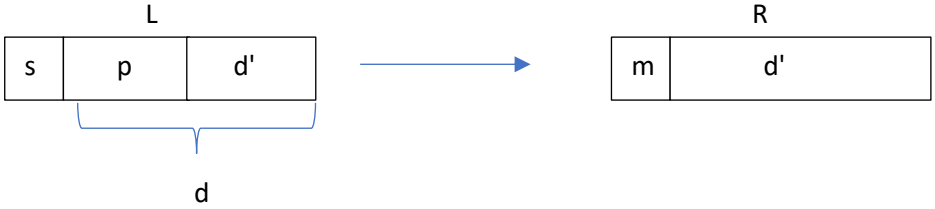
EN PAGINACIÓN

- Tamaño en bits de L
- Tamaño en bits de R
- Tamaño en bits de p
- Tamaño en bits de d
- Tamaño en bits de m

EN SEGMENTACIÓN

- Tamaño en bits de L
- Tamaño en bits de R
- Tamaño en bits de s
- Tamaño en bits de d

EN SEGMENTACIÓN PAGINADA



Tamaño en bits de L

Tamaño en bits de R

Tamaño en bits de s

Tamaño en bits de p

Tamaño en bits de d

Tamaño en bits de d'

Tamaño en bits de m

Ejemplos de ejercicios

Ejercicio 1

Un SO utiliza un esquema de memoria de segmentación. Se sabe que las direcciones lógicas son de 10 bits y que las direcciones físicas son de 16 bits. Además, se sabe que el máximo tamaño de un segmento es 256 bytes. Si un proceso tiene dos segmentos:

- Segmento 0 de tamaño 4 bytes
- Segmento 1 de tamaño 20 bytes

Indíquese, en el proceso anteriormente mencionado, qué dirección lógica (en decimal) va a continuación de la dirección lógica 3

Resolución:

Datos que tengo que sacar del enunciado:

La L me la dan directamente y son 10 bits

La R me la dan directamente y son 16 bits

La d es el número de bits que se reservan para desplazamientos de segmento. Si el máximo tamaño de un segmento es 256 entonces se necesitan 8 bits ($\log_2 256 = 8$). Por tanto, la d es 8 bits.

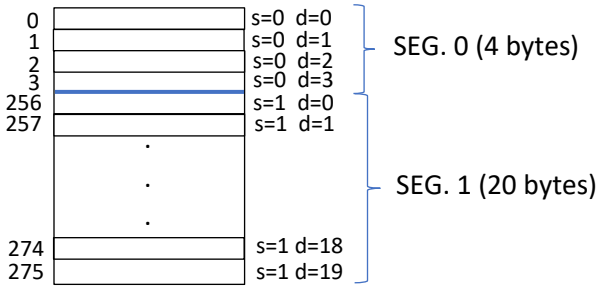
Una vez se tiene L y d la s es trivial porque es la diferencia y por tanto son 2 bits (10 – 8 = 2)

Con esta información previa ya puedo enfocar el ejercicio. Lo primero es pasar la dirección lógica a binario y representarla en 10 bits, luego divido 2 + 8 bits y obtengo el segmento y el desplazamiento de dicha dirección.

3 en binario es 11. Como tienen que ser 10 bits relleno con 0 por la izquierda. Así, la dirección lógica 3 del proceso es, en binario:

s d
00 00000011

Por tanto, la dirección lógica 3 del proceso es el segmento 0 y desplazamiento 3. Como el segmento 0 tiene exactamente 4 bytes quiere decir que su último desplazamiento es el 3. Es decir, esta dirección lógica es justamente la última dirección del segmento 0 y por tanto la dirección siguiente estará ya en el siguiente segmento (desplazamiento 0). Veamos esto con un dibujo del proceso:



La dirección siguiente a la dirección lógica 3, que es el segmento 0 desplazamiento 3, es la que corresponde al segmento 1 desplazamiento 0, en binario:

01 00000000

Si pasamos ese número en binario a decimal nos sale **la dirección lógica 256**.

Ejercicio 2

En el ejercicio anterior, razónese ahora por qué no es válido comprobar si una dirección es inválida de la misma forma que en el resto de los esquemas, es decir **L < Tamaño proceso**

Resolución:

Lo primero es calcular el tamaño en bytes del proceso que es 4 bytes del primero segmento más 20 bytes del segundo, es decir 24 bytes.

Si, por ejemplo, comprobáramos si la dirección lógica 256 es válida, y lo hacemos igual que el resto de esquemas de memoria, sería:

¿Es 256 < 24? Y nos daría error de dirección inválida.

Sin embargo, nosotros sabemos, por el dibujo de arriba, que la dirección lógica 256 sí es una dirección del proceso y debería ser válida.

Este es el motivo por el cual no vale la protección clásica. En su lugar se deben comprobar dos cosas:

1. Que el segmento de la dirección lógica exista (que **s** tenga entrada en la tabla de segmentos).
2. Que el desplazamiento dentro del segmento no exceda al tamaño del segmento (que **d** < Tam. Segmento)

Ejercicio 3

Si el bus de direcciones de una máquina tiene 20 líneas y el sistema operativo, que utiliza un modelo de memoria de paginación simple, divide a la memoria principal en 65536 marcos de página, indíquese el formato de las direcciones lógicas y el de las direcciones físicas.

Resolución:

Datos que tengo que sacar del enunciado:

Si el bus de direcciones tiene 20 líneas entonces las direcciones reales (R) son de 20 bits.

Si el SO divide la MP en 65536 marcos de página entonces nos están dando indirectamente la m. Ya que m serán tantos bits como necesite para almacenar 65536 números de marcos, es decir 16 bits ($\log_2 65536=16$).

Ahora es trivial calcular **d** ya que será la diferencia entre **R** y **m**, es decir 4 bits (20-16=4)

Ya solo me falta calcular **p**. El problema es que ya no me dan más información en el enunciado y aparentemente no se puede calcular. No es cierto. Si no nos dicen nada acerca del tamaño de **L** (tamaño de las direcciones lógicas) deberemos asumir que el SO permite cargar procesos lo más grande posible y eso sería que **L=R**. Por tanto, **L** son también direcciones de 20 bits.

Ahora es trivial calcular **p** ya que será la diferencia entre **L** y **d**, es decir 16 bits (20-4=16).

La respuesta final sería: