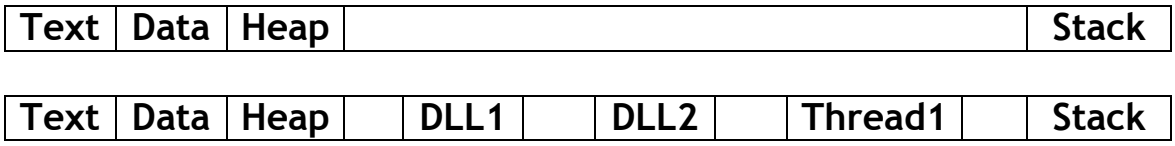
Consideraciones de la Tabla de Páginas

Las aplicaciones requieren espacios de direcciones crecientes, con más fuerza que la propia evolución de la tecnología de memoria. Se puede agregar a esta consideración que la evolución en los chips de memoria no es continua, sino por escalones. Dada la organización con un direccionado en dos dimensiones, como veremos más adelante, la incorporación de un bit más en cada dimensión motiva que de una generación a otra se pasa cuadriplicando la capacidad anterior.

Esta demanda creciente de espacio de direccionado se soporta por ejemplo pasando de una arquitectura de **32 bits** a arquitecturas de **64 bits** que con *sistema paginados* posibilitan un *espacio lógico* de **16TB**. Pero claramente no es una condición necesaria. El trabajar con *sistemas segmentados* potencia el espacio lógico alcanzable por la multiplicidad de espacios de este esquema. Así, por ejemplo, en el Pentium, si adoptamos el esquema de segmentado se combina un especio lineal de **32 bits** con los **14 bits** que seleccionan el segmento en cuestión, otorgando un espacio total de **64TB** producto de los **66 bits** de dirección lógica.

Otra cuestión, que ya se ha mencionado, es que cuando a través de sistemas paginados soportemos *librerías de run-time* y *procesos multithreading*, prácticas que actualmente son comunes, se requiere soporte para un especio de dirección disperso, con huecos. Estos huecos son necesarios para permitir crecimiento dinámico sin colisión en el especio lógico. Esto en contraste con lo que ocurriría con un sistema tradicional como el **Unix 4.3 BSD** el cual estaba compuesto por sólo dos regiones contiguas.



En ese caso no es necesario ocuparse del espacio libre entre ambos extremos. En cambio las librerías y los threads se esparcirán por este espacio, con lo cual la tabla de páginas deberá contemplarlo en su totalidad, esto es no podrá saltearlo.

Esta situación, es decir, el requerimiento sobre la tabla de páginas, introduce dos cuestiones:

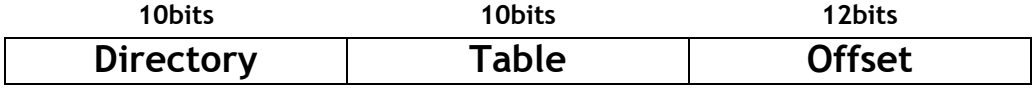
- Mucho recurso de memoria empleado para la tabla de páginas.
- El problema de la *fragmentación externa* a nivel de la tabla, en el sentido que es un área contigua a la cual se indexa con el **VPN**. Se requiere un espacio contiguo de tamaño suficiente.

¿Cómo se atienden estas cuestiones? Lo del uso del recurso de memoria se maneja permitiendo que la tabla de páginas se encuentre paginada, esto es, parte en memoria y parte en disco. La cuestión del espacio contiguo queda resuelta organizando la tabla de páginas de forma *jerárquica*, no en un único nivel.

A manera de ejemplo de lo anterior podemos mencionar la **VAX** en donde las tablas de páginas de los procesos se desarrollaban en el espacio lógico del sistema. Hablaremos luego de contigüidad en el espacio lógico del sistema, no físico. De tal forma requiere de una doble traslación, se agrega la destinada a obtener la dirección de memoria en donde ubicar el *frame pointer* dado que el puntero a la tabla de páginas del usuario **UPTP** (User Page Table Pointer) es una dirección lógica. Además, en ese acceso a la tabla de páginas del usuario, se podría

producir un *page fault*, asociado a que la porción de la tabla de páginas que contiene el frame pointer no se encuentra en memoria principal, se encuentra en disco.

Otro ejemplo lo encotramos en el **Pentium II**, en donde el espacio lineal (aquél que corresponde a un segmento) en caso de estar paginado se organiza en tres campos. El primero de *directorio de página*, el segundo de *tabla de página* y el tercero de *offset*. Dado un segmento de un proceso este tendrá la dirección de la tabla de directorio. Indexando ésta con los primeros **10 bits** del campo PAGE obtendremos el frame pointer.



Se puede observar que la distribución de bits no es casual. Esta acorde con el tamaño de página de **4KB**, dado que cada entrada en las tablas es de 4 bytes, con lo cual tanto la tabla de directorio como la de página ocupan una página completa, habiendo desaparecido obviamente el requerimiento de espacio contiguo para la tabla de páginas. Más aun, mientras que la tabla de directorio estará siempre residente en memoria, cada una de sus entradas dispondrá de un bit **P** de presente. Para el caso de esta tabla dice que la tabla apuntada está o no en memoria (parte en disco); análogamente, el mismo bit **P** en una entrada de la tabla de páginas indicará si la página está o no presente en memoria.

Otra solución a los dos temas planteados alrededor de la tabla de páginas es el empleo de una *tabla de páginas invertida*. De lo que se trata es de englobar en una única estructura la información de todas las páginas, sin distinción de proceso, residentes en memoria. Luego con alguna técnica de *hash* se realiza el almacenamiento y búsqueda habida cuenta que se debe realizar una búsqueda asociativa (no como antes con búsqueda directa). La tabla estará acotada en tamaño por el tamaño de la memoria principal del sistema y el tamaño de página, no existiendo tampoco el problema de manejar áreas contiguas.

Para todas las soluciones vistas se puede argumentar que las búsquedas dan lugar a procesos relativamente lentos. Si bien esto es cierto, no se debe olvidar que estos actúan en segunda instancia, el objetivo es manejar tasas de acierto altas en el *buffer de traslación* **TLB**. El problema que enfrentamos en sistemas paginados es que el tamaño de página como sabemos es una decisión del diseñador, con un tamaño típico de **4KB** a **8KB**. Hay aplicaciones que tienen la característica de referenciar áreas de memoria muy vastas, como ser tablas del sistema operativo o en aplicaciones gráficas. Esto, dada la granularidad relativamente pequeña de las páginas, induce el empleo de numerosas páginas, lo que contaría la localidad espacial prevista y provoca una consecuente pérdida de eficiencia en el **TLB**. Nótese que esto no ocurre con sistemas segmentados, dado que como se dijo el tamaño de éstos se ajusta a la aplicación. Usar *superpáginas* o *superbloques* son las alternativas para solucionarlo. Intel por caso en las entradas de la tabla **Directory** tiene un bit **PS** (tamaño de página) que le indica si se trata de una página normal o bien una superpágina de **4MB**. De tal forma se reducirá el número de entradas al **TLB**. Para direccionar usamos los bits superiores (esto es, la dirección de la tabla de página sin los **10 bits** inferiores) para la dirección base de la superpágina y los bits del campo **Table** y **Offset** en conjunto dan el desplazamiento, esto es, resultan **22 bits** para **4 MB**.