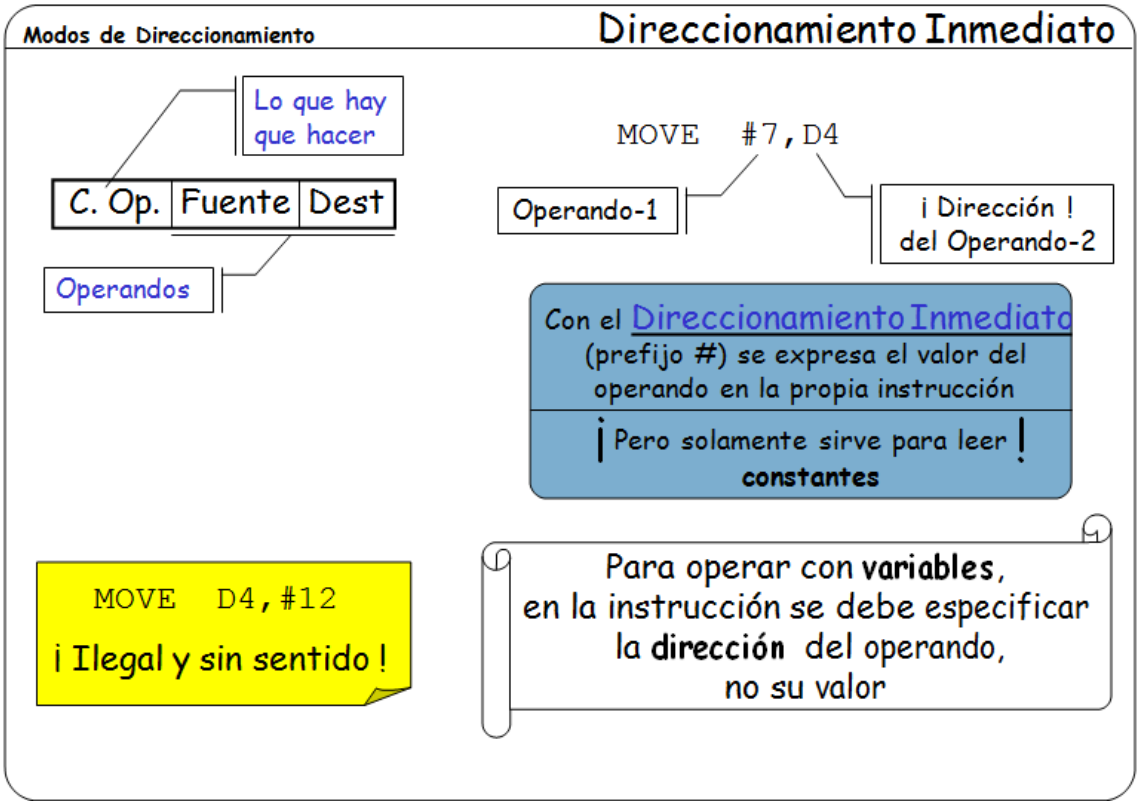
MODOS DE DIRECCIONAMIENTO

Adrián González Rodrigo

José Manuel Gómez Pérez

Daniel Sánchez Cerrón



El formato de una instrucción máquina consta de código de operación y de operandos (es una costumbre aceptada en informática el considerar al resultado o destino de la operación también como un operando).

Si el código de operación indica lo que hay que realizar, cada operando debería contener su valor.

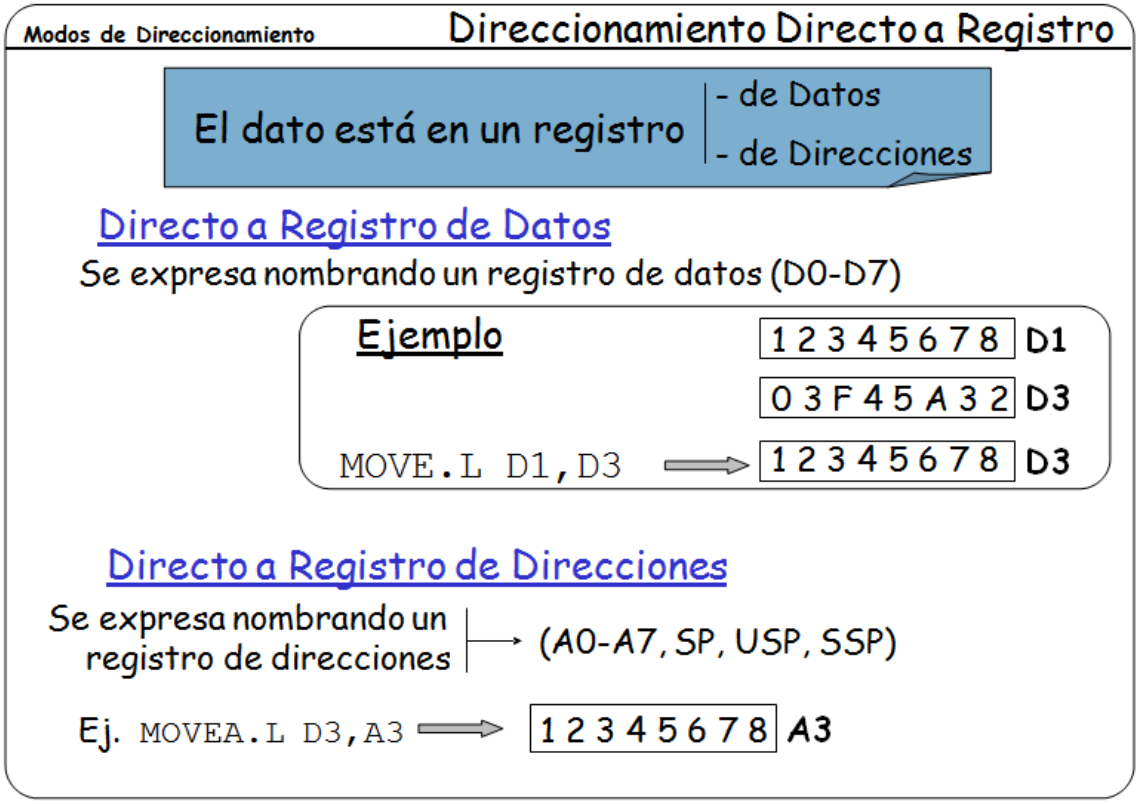
Así tenemos en el ejemplo una instrucción que copia el valor 7 al registro D4. En esta situación en la que **el valor del operando está en la propia instrucción** (el 7, en nuestro ejemplo), decimos que se tiene un **direccionamiento inmediato** al operando.

Pero si nos fijamos, en el operando de destino no está su valor, sino la dirección donde hay que dejar el operando fuente (D4 en nuestro ejemplo).

Se puede observar que con el direccionamiento inmediato el valor del operando es siempre el mismo (es una constante), puesto que se escribe en la propia instrucción en tiempo de compilación. Entonces ¿qué pasa si queremos operar con variables?

Está claro que **el direccionamiento inmediato no sirve para operar con variables**. En su lugar lo que se necesita es especificar la dirección de memoria donde está la variable, la cual tendrá valores distintos en distintas circunstancias.

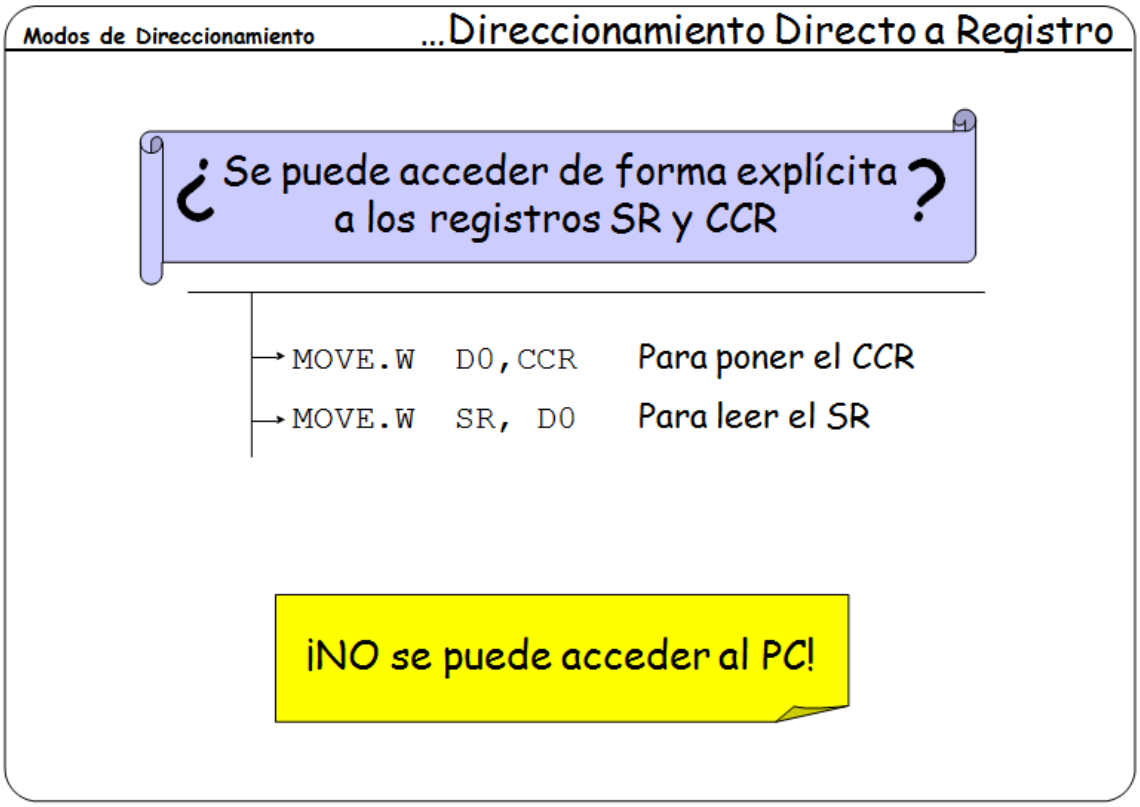
Hay diversos mecanismos o "modos de direccionamiento" para indicar la dirección de un operando en el 68000. Veámoslos a continuación.



Con direccionamiento directo a registro, el valor del operando se encuentra en uno de los registros internos del 68000. Por esto, en el campo de operando simplemente debe indicarse el nombre de tal registro.

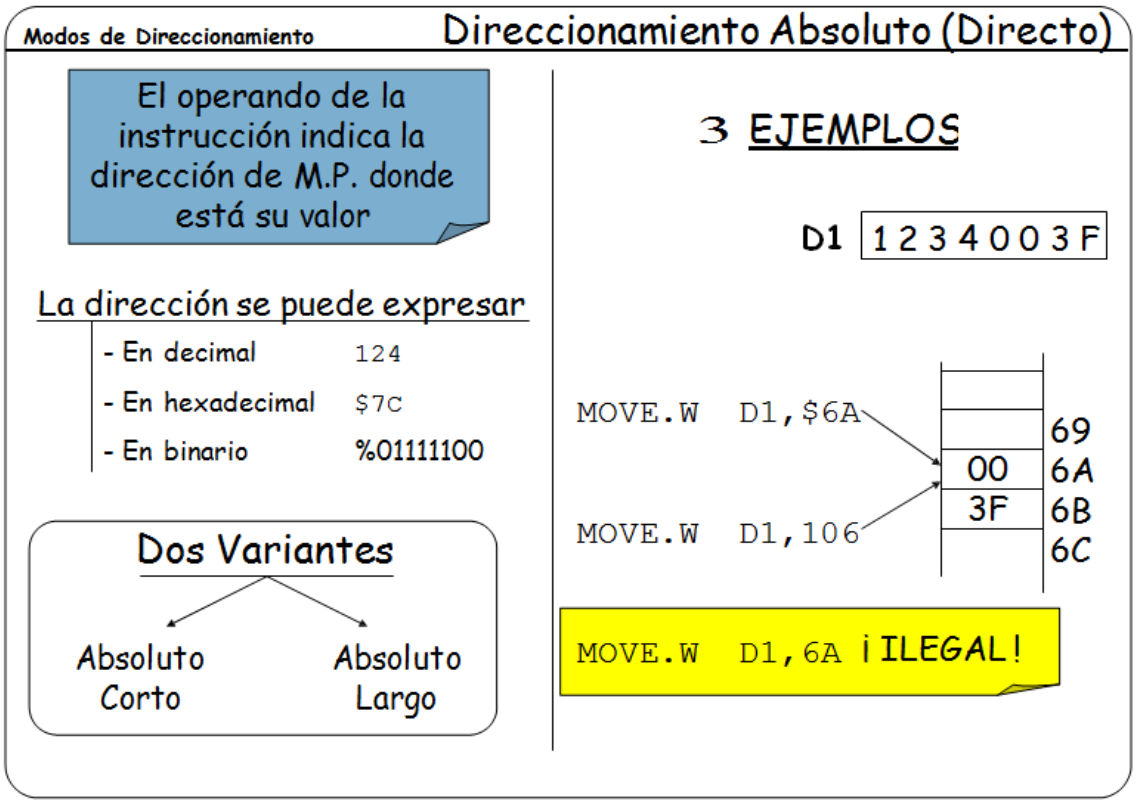
Con este direccionamiento se puede acceder tanto a operandos fuente como de destino o de resultado, y como no implica accesos a memoria principal, resulta un modo rápido de direccionamiento.

Como registros de operandos puede utilizarse registros de datos (D0-D7) o de direcciones (A0-A7).



En algunas instrucciones también se permite acceder al registro de estado en su conjunto (SR), o solamente al CCR, aunque algunos de estos accesos solamente se permiten en modo supervisor.

No hay ninguna instrucción para acceder explícitamente al contador de programa, aunque implícitamente todas las instrucciones lo modifiquen para poder extraer la siguiente instrucción de memoria.

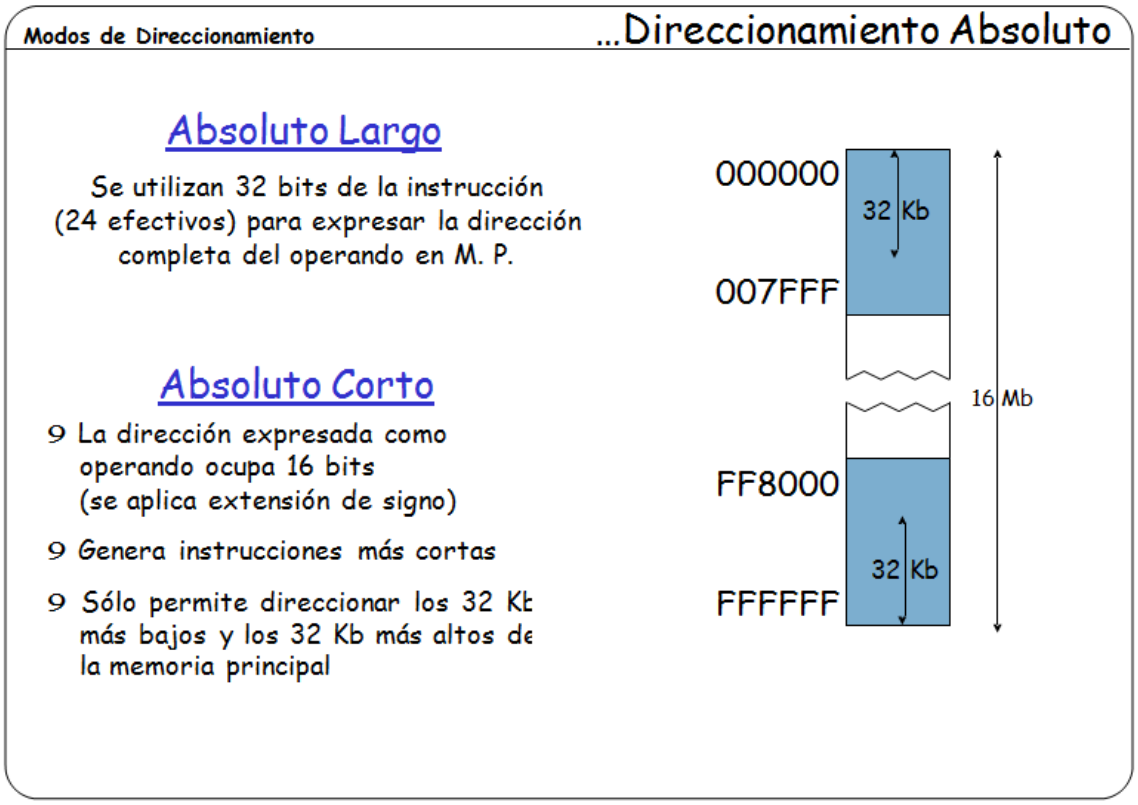


A este modo de direccionamiento se le conoce como direccionamiento **absoluto**, **directo a memoria**, o simplemente **directo**. En este caso, en el campo de operando de la instrucción se indica la dirección de memoria principal donde se encuentra el valor del operando o su destino.

Este modo también es conocido como "absoluto" debido a que la dirección de la variable debe conocerse en tiempo de compilación y no varía nunca. Este direccionamiento solamente es válido para programas no reubicables, o bien para programas que siendo reubicables utilizan este direccionamiento solamente para acceder a dispositivos periféricos cuyas direcciones (siempre fijas) están *mapeadas* o representadas en el espacio de direccionamiento de la memoria principal.

Como se puede ver en los ejemplos, para denotar un operando en ensamblador, simplemente hay que poner una dirección (prefijado por **$** para base hexadecimal o **%** para base binaria) o la etiqueta de la variable.

Este direccionamiento tiene dos variantes, dependiendo del número de bits que se requieren en el campo de operando de la instrucción para indicar la dirección. Veámoslos en la siguiente transparencia.



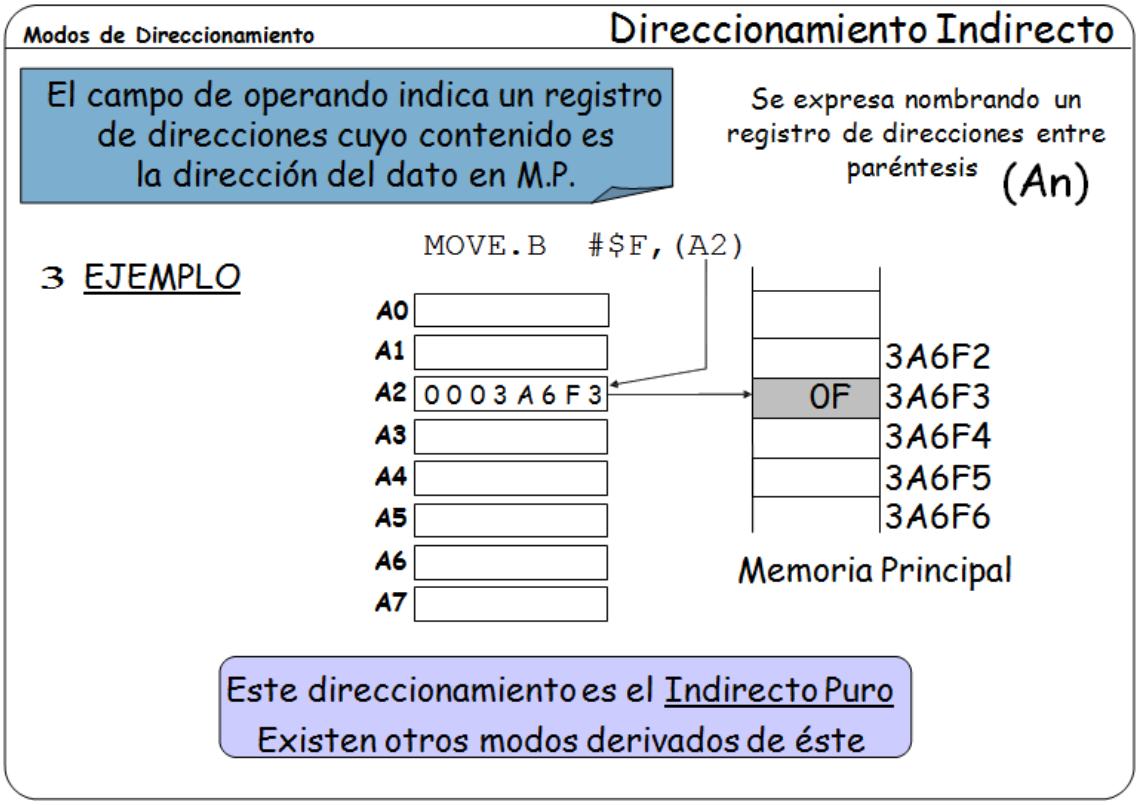
En el **direccionamiento absoluto largo**, el campo de operando consta de 32 bits en los que se indica la dirección absoluta y completa del valor del operando. Con los 32 bits se permite especificar cualquier área de memoria de todo el espacio de direccionamiento del 68000.

Con el **direccionamiento absoluto corto**, el campo de operando está formado por solo 16 bits. A partir de esta palabra se forma una dirección de 24 bits con extensión de signo. Con 16 bits con signo solamente puede accederse a dos áreas de la memoria:

$000000 - $007FFF (los 32 primeros Kbytes)

$FF8000 - $FFFFFF (los últimos 32 Kbytes)

A pesar de existir dos variantes del direccionamiento absoluto, el programador de ensamblador no tiene que preocuparse de esto, simplemente debe indicar el nombre de la variable o la dirección, pues el ensamblador selecciona automáticamente el modo más conveniente en cada caso.



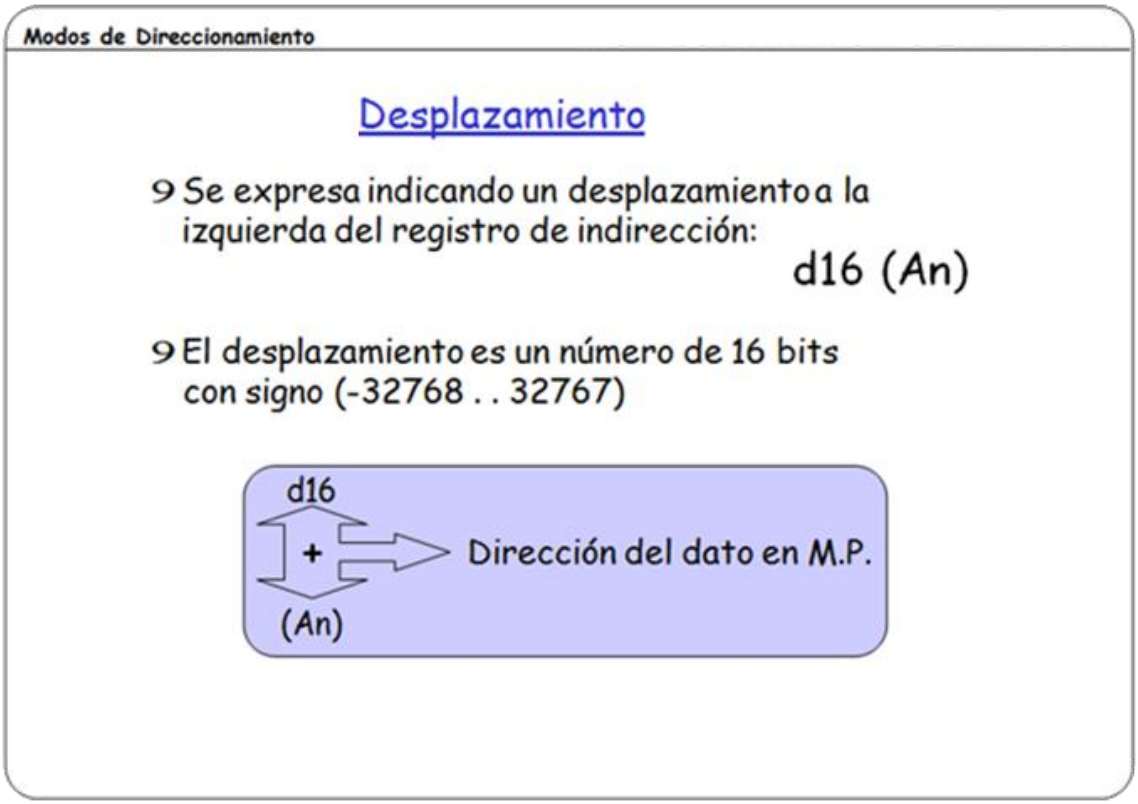
Direccionamiento indirecto por registro significa que la dirección del operando se encuentra en un registro. En el 68000, son los registros de direcciones los que se ocupan de esto (A0-A7).

Obsérvese que si en el direccionamiento absoluto el operando era una variable, pero su dirección era fija y constante por encontrarse en la propia instrucción, en el direccionamiento indirecto también es variable la dirección, pues ésta se encuentra en un registro de direcciones.

Como se muestra en los ejemplos, la indirección se denota encerrando entre paréntesis el registro que contiene la dirección de la variable. An hace referencia al contenido de An, mientras que (An) hace referencia al contenido de la dirección de memoria apuntada por An.

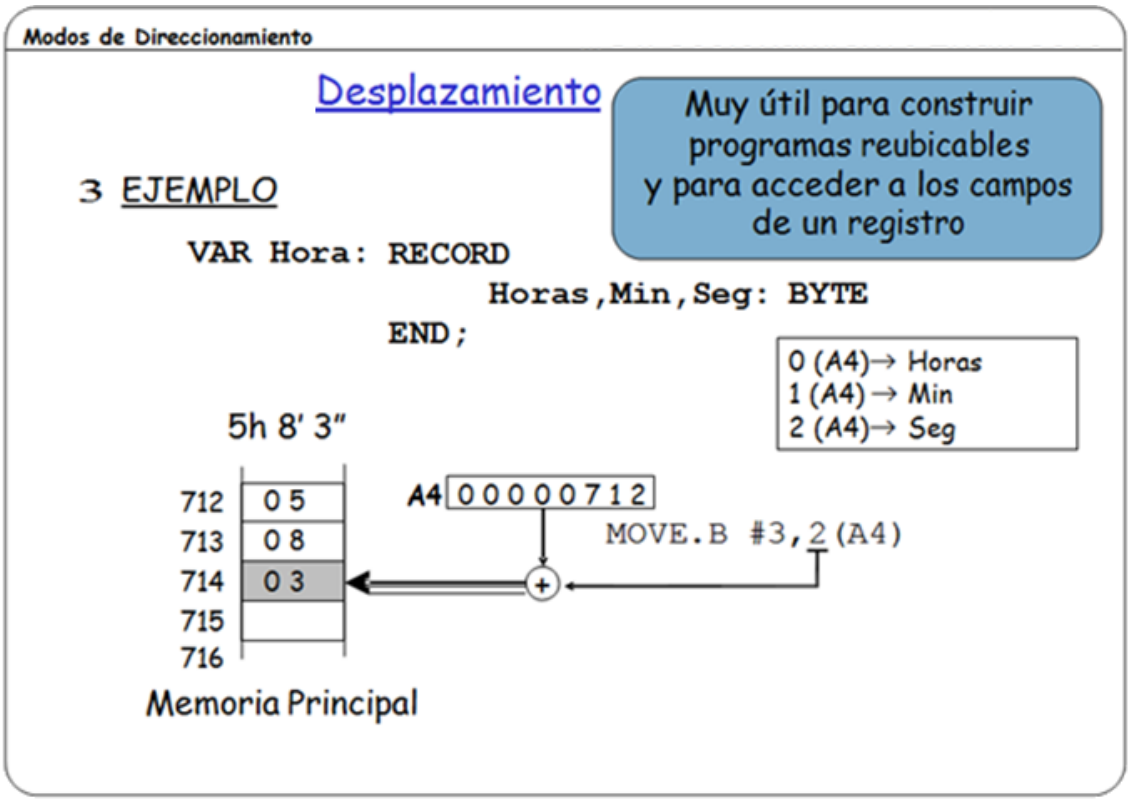
Podemos ir adelantando que los registros de direcciones suelen cargarse con instrucciones como `MOVEA` (*Move Address*), o `LEA` (*Load Efective Address*).

Esto que hemos comentado es el “direccionamiento indirecto por registro simple”. Pasemos a ver algunas variantes.



Como se puede ver en la transparencia, el **direccionamiento con desplazamiento** es muy similar al indirecto puro. En este caso, la dirección efectiva se obtiene no solo con el contenido del registro de indirección, sino añadiéndole además un número de 16 bits con signo (el desplazamiento) contenido en la propia instrucción (como una constante). Esto significa que a partir de una dirección cualquiera de memoria contenida en el registro de indirección, el desplazamiento puede modificar tal dirección en +- 32 Kbytes.

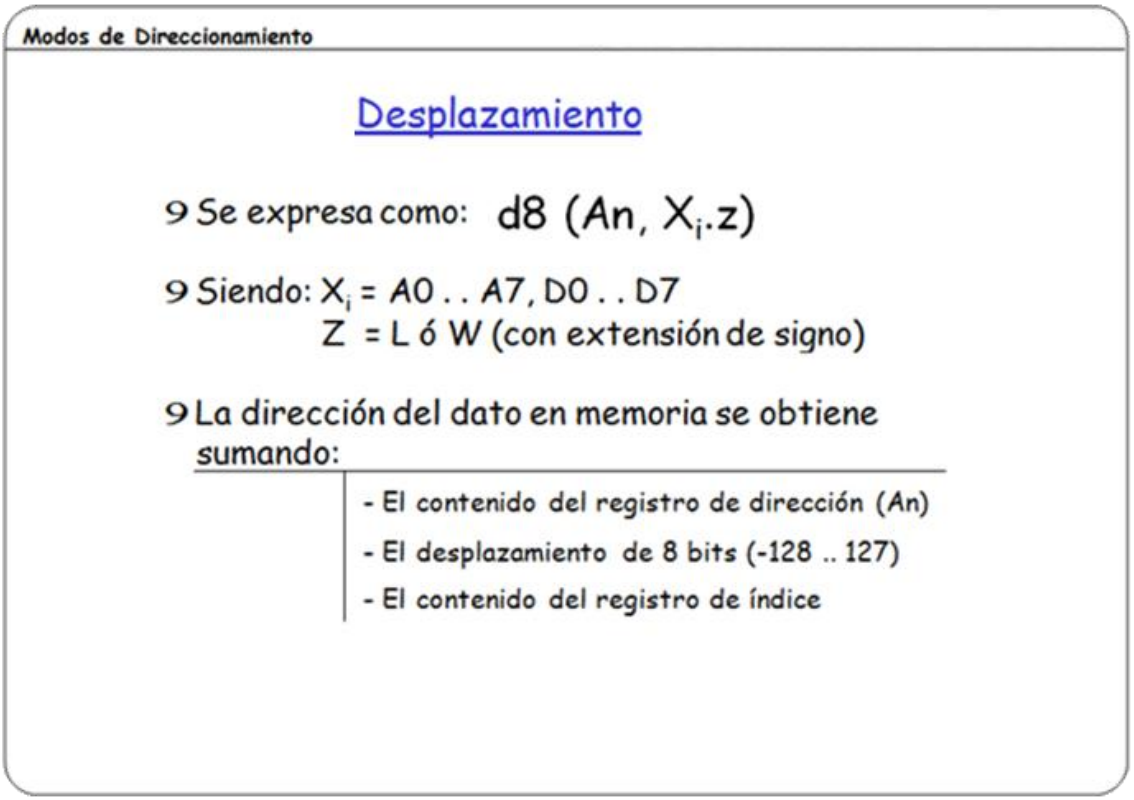
Obsérvese que para sumar un número de 16 bits con signo a otro de 32 bits, también con signo, al número de 16 bits se le aplica previamente una extensión del signo hasta los 32 bits.



El direccionamiento con desplazamiento **es útil para la escritura de programas reubicables**, de tal manera que el registro se carga, en ejecución, con la dirección del comienzo de los datos, y los accesos a los operandos se calculan sumando a la dirección de comienzo de los datos (que es variable) el desplazamiento relativo (que es constante y establecido en tiempo de compilación).

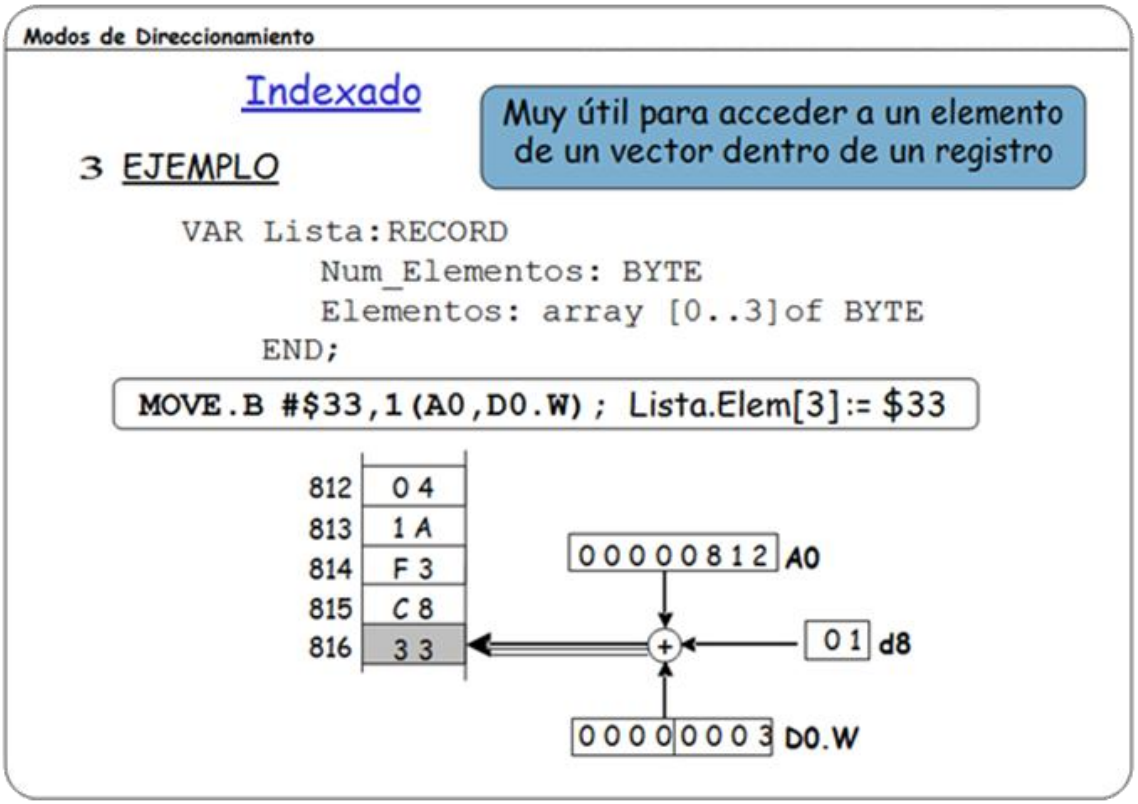
Otra aplicación (también reubicable) consiste en cargar en el registro la dirección de una estructura de datos (como un registro de Pascal) y con el desplazamiento hacer referencia a cada uno de sus campos.

Aquí mostramos un ejemplo con un registro que contiene la hora (hora, minutos y segundos) y en el que cada uno de sus campos ocupa un byte. Con la direción del comienzo del registro (712) cargada en el registro A4, se accede a cada uno de sus campos con los desplazamientos 0, 1 y 2.



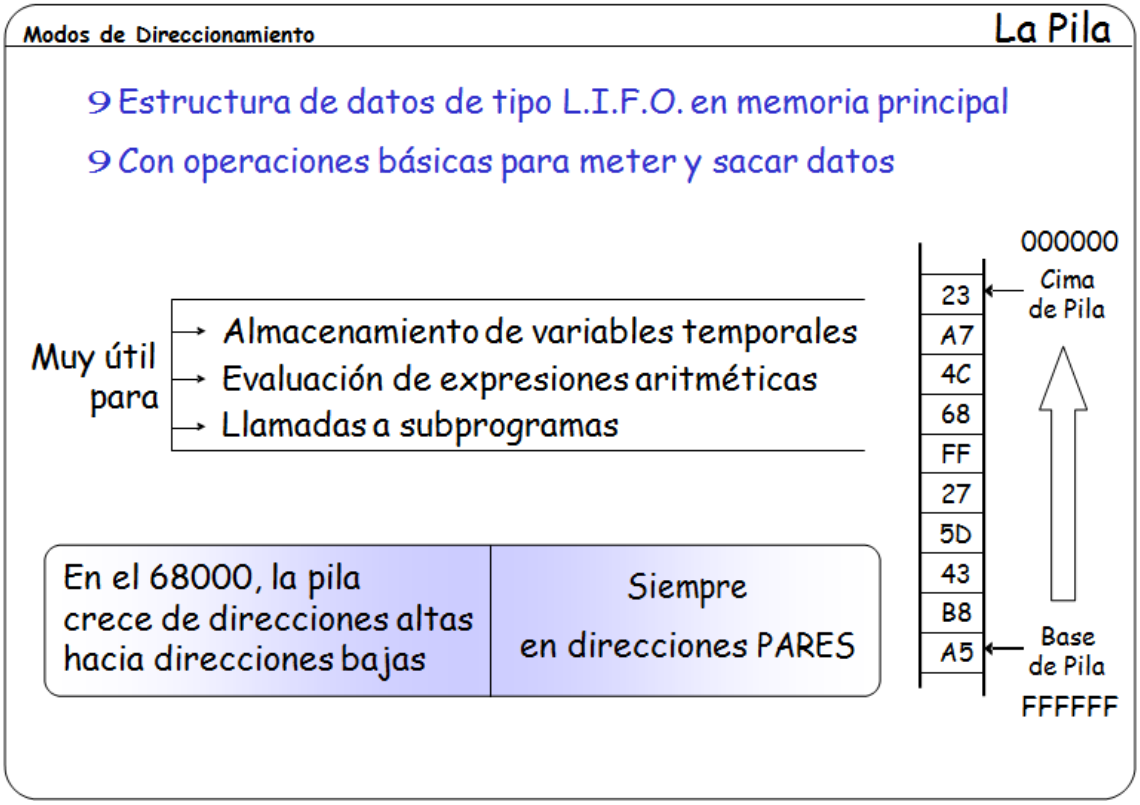
El siguiente paso en el direccionamiento indirecto nos lleva al **direccionamiento indexado**. Es similar al indirecto con desplazamiento, pero en el caso indexado se dispone de un registro más (registro de índice X_i) cuyo contenido también debe sumarse en el cálculo de la dirección efectiva. Obsérvese que el registro de índice puede ser uno cualquiera de datos o direcciones, y que mediante el sufijo ".W" o ".L" puede considerarse la totalidad del registro o solamente la palabra de menor peso.

Como se observa en la transparencia, cuando el direccionamiento indirecto utiliza registro de índice, el desplazamiento es un número, con signo, de solo 8 bits, lo que quiere decir que permite una modificación constante en un rango de -128 a +127.



La utilidad de este direccionamiento está en la gestión de tablas de dos dimensiones, de vectores donde cada elemento de la tabla es un registro o estructura de datos, o para acceder a los elementos de un vector que es un campo de un registro.

Por ejemplo, si tenemos un vector cuyos elementos son estructuras de datos (registros), para acceder a los campos de un registro concreto, en el registro de indirección `An` se cargaría la dirección de comienzo del vector y en el registro de índice, el desplazamiento hasta un registro concreto (un elemento del vector). Sumando el desplazamiento de 8 bits se obtendría la dirección completa del campo deseado.

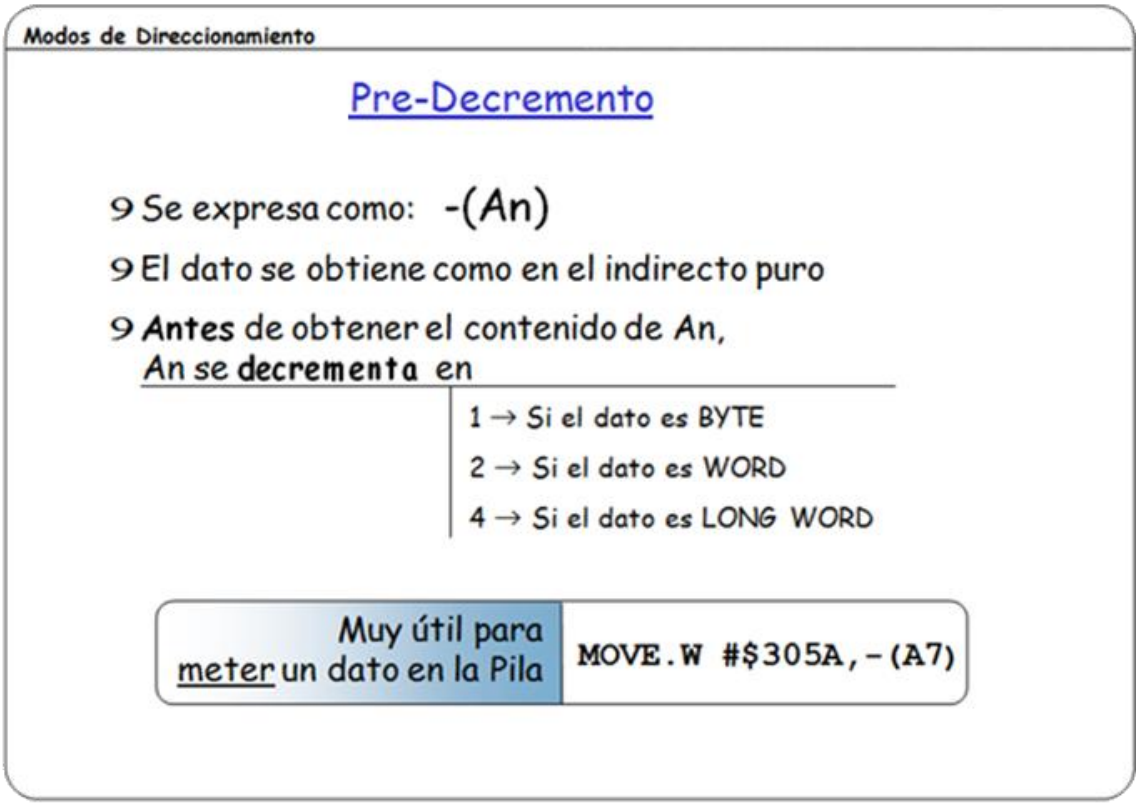


La Pila es una estructura de datos que se mantiene en memoria principal. Básicamente es un vector de celdas de datos de direcciones secuenciales, en los que se meten y sacan datos con política LIFO (*Last In, First Out*), es decir, el último en entrar es el primero en salir.

Esta estructura es muy útil como ayuda en la ejecución de los programas, pues se utiliza para guardar en ella variables temporales, parámetros, direcciones de retorno en la llamada a procedimientos, etc.

El primer elemento que se mete en la pila se dice que está en el fondo de la pila, y el último que se ha metido está en la **cima de la pila**. El interés de la cima de la pila es que siempre está en ella el primer dato disponible de la pila. Las operaciones con la pila son **meter** y **sacar** datos de ella, y operan utilizando un registro denominado **puntero de pila** que siempre contiene la dirección de la cima de la pila.

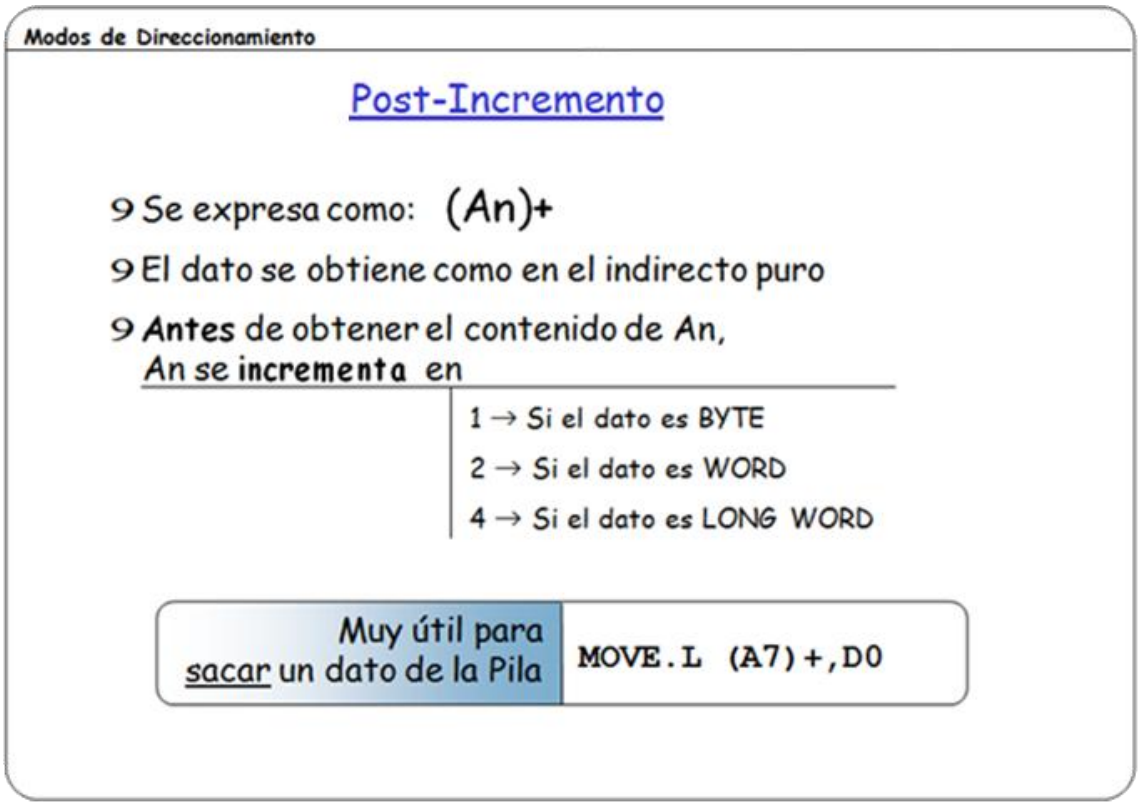
En Motorola, la Pila es una estructura de datos que, quizás en contra de la intuición, crece de las direcciones altas hacia las bajas, es decir, que la cima de la Pila siempre está en una dirección más baja que la base de la Pila (salvo cuando la Pila está vacía, claro).



Las dos últimas variantes del direccionamiento indirecto son las que permiten modificar el contenido del registro de indirección antes o después del cálculo de la dirección efectiva.

El **direccionamiento con pre-decremento** le resta al registro de indirección el tamaño del dato al que se accede, de tal manera que si la operación lleva el sufijo ".B", se resta 1; si el sufijo es ".W", se le resta 2 al registro, y si el sufijo es ".L", se le resta 4. Una vez realizada la resta, se toma el contenido del registro como la dirección del operando. Esto quiere decir que si antes de la ejecución de esta instrucción, el registro de indirección An apuntaba a una cierta dirección de memoria, después de comenzar su ejecución, y justo antes de acceder al operando, el registro apunta a una dirección de memoria 1, 2 o 4 bytes más baja.

El registro A7 (o puntero de pila) supone una excepción, pues cuando se utiliza como registro de indirección con predecremento o postincremento, se incrementa o decrementa en 2 tanto para un operando de tipo *word* como para un byte, y en 4 para las dobles palabras.

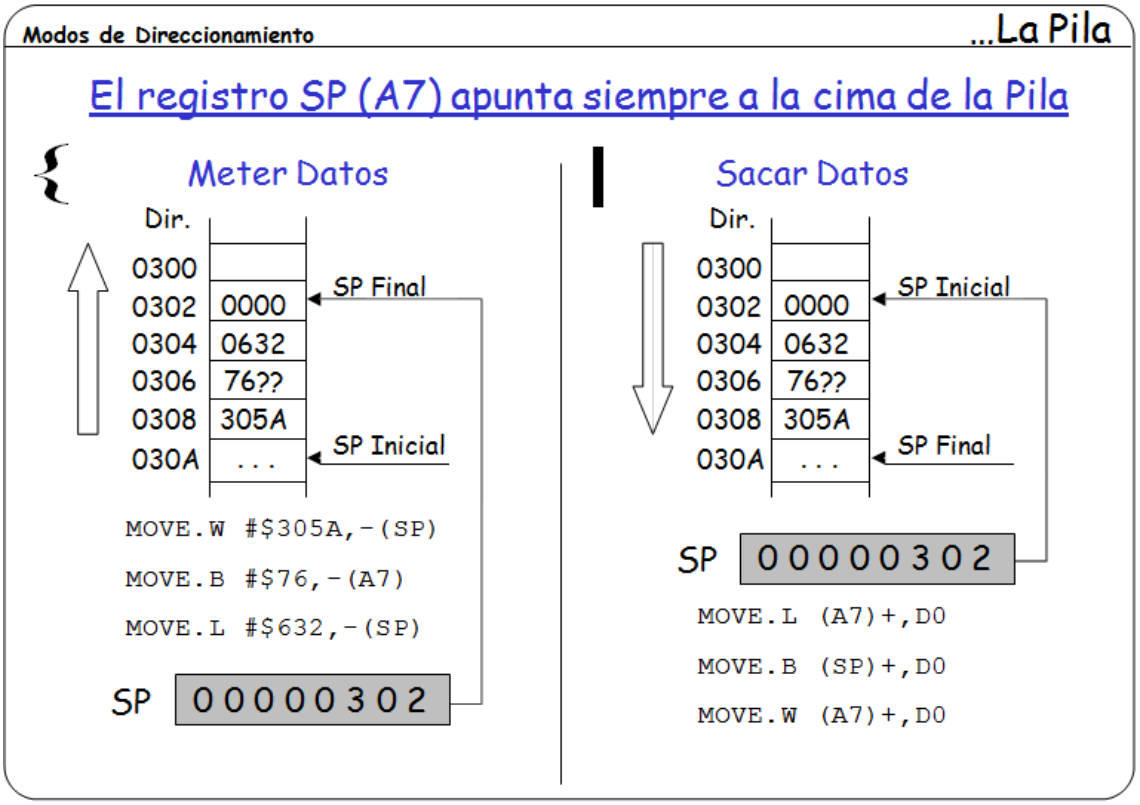


El **direccionamiento con post-incremento** se comporta de manera análoga al caso anterior, con la diferencia de que ahora, en primer lugar se toma el contenido del registro de indirección como la dirección del operando, y a continuación se le añade a dicho registro un valor igual al tamaño del operando.

Aunque pueda parecer un poco rebuscado, este direccionamiento es fundamental para la gestión de estructuras de datos de tipo "pila", muy comunes entre los procesadores actuales. Pasemos a la siguiente transparencia para comentar brevemente la utilidad de la pila y cómo se gestiona fácilmente con los dos modos de direccionamiento que acabamos de ver.

Estos dos direccionamientos también resultan útiles para tratar con vectores de datos cuyos elementos están almacenados consecutivamente en memoria. Por ejemplo, para sumar los *n* elementos (de una palabra de tamaño) de una tabla cuya dirección de comienzo está contenida en A0, basta con ejecutar *n* veces la siguiente instrucción:

```
ADD.W (A0)+,D0
```



En el 68000, el puntero de pila (SP) puede ser cualquier registro de direcciones, pero algunas instrucciones que acceden implícitamente a la pila, como las de llamada y retorno de subrutina utilizan el registro A7, por lo que éste es el que se utiliza normalmente.

Así, el acceso a la pila se realiza mediante instrucciones de movimiento de datos y el ya comentado direccionamiento indirecto por registro. En la pila del 68000 los elementos de la pila ocupan siempre direcciones pares. Cuando se mete un byte en la pila, se deja inutilizado el byte adyacente para mantener las direcciones pares.

Sabiendo ya cómo es la pila del 68000 y sus operaciones, resulta fácil ver que el direccionamiento indirecto con pre-decremento es el apropiado para meter elementos en la pila, mientras que el indirecto con post-incremento se utiliza para sacar datos de ella.

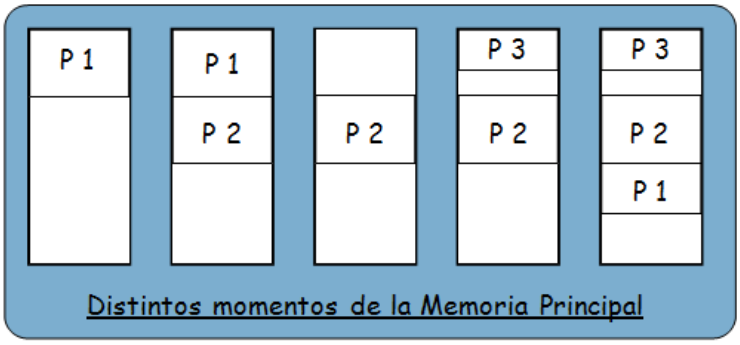
Obsérvese el efecto de las operaciones de meter datos en la pila ({) partiendo de un Puntero de

Pila con el valor 030A. Las operaciones de sacar datos (▌) comienzan con un Puntero de Pila con el valor que dejó la última operación de meter datos, o sea, 0302.

Modos de Direccionamiento

Relativo al PC

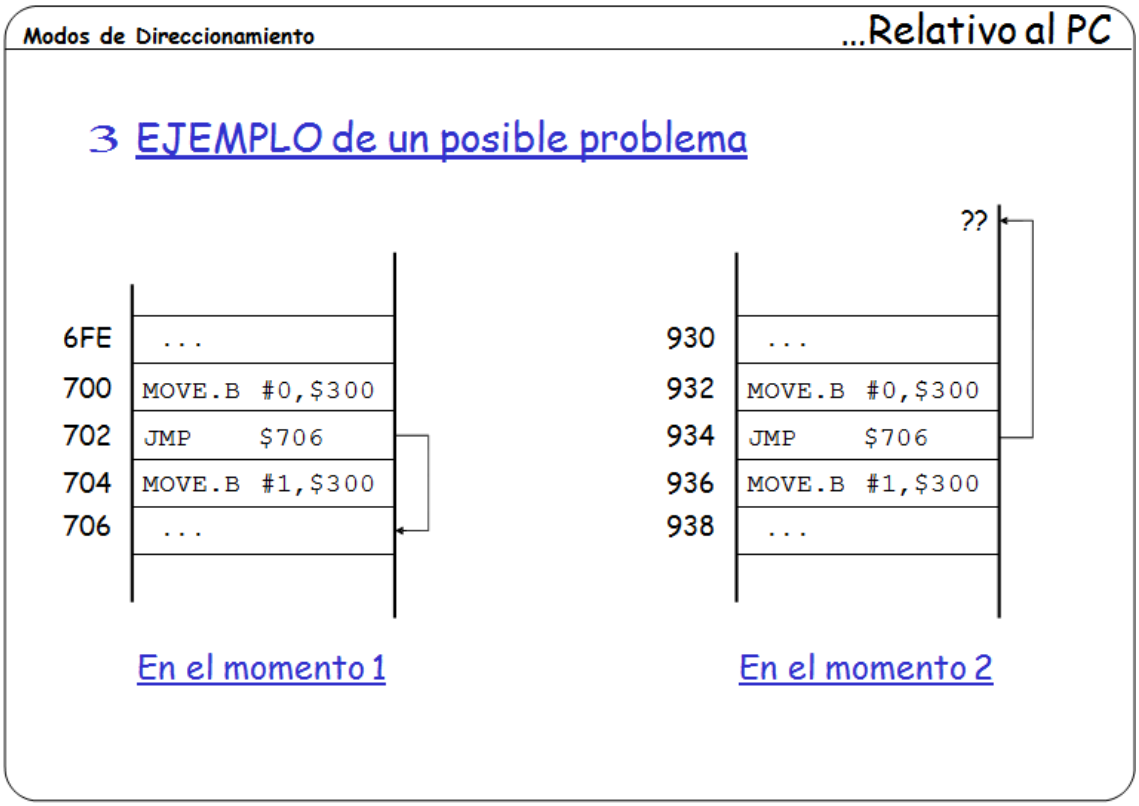
- En un sistema multitarea puede existir más de un programa cargado en la Memoria Principal (ejecutándose)
- En tales sistemas, un mismo programa puede cargarse en zonas distintas de memoria cada vez que se ejecuta



¡ESTO PUEDE SER UNA FUENTE DE PROBLEMAS!

En los sistemas multiusuario y en casi todos los sistemas operativos en general, suele ser normal la multitarea, según lo cual múltiples programas están cargados en memoria ejecutándose. Por esto, cuando se solicita la ejecución de un cierto programa no se sabe a priori el área de memoria que va a estar disponible para cargarlo.

Teniendo en cuenta esto, ya hemos visto que ciertos tipos de direccionamiento, como el absoluto o directo, no son apropiados para estos sistemas. También hemos visto que el direccionamiento indirecto soluciona bien el acceso a las variables.



Como ya veremos, hay unas instrucciones de salto que alteran el flujo de ejecución secuencial de los programas, y en estas instrucciones se debe indicar la dirección de la siguiente instrucción a ejecutar. En ensamblador, esta dirección se suele indicar mediante una etiqueta que identifica la dirección de una instrucción.

Pues bien, la dirección de una instrucción a la que se desee saltar desde algún punto del programa también es variable, es decir, depende de la zona de memoria en la que se cargue el programa.

Como vemos en la diapositiva, en el *Momento 1* es posible que un programa esté cargado en una dirección de memoria tal que una instrucción de salto a la dirección $706 se ejecute tal y como desea el programador. Sin embargo, en otro *Momento 2*, si este mismo programa se carga en otra dirección distinta, como se puede apreciar, la instrucción de bifurcación a la dirección $706, no va a saltar a la instrucción deseada (la de la dirección $938), sino a la instrucción de la dirección $706; dirección que quizás cae incluso fuera del espacio del programa.

Como vemos, los saltos a direcciones absolutas suponen una pega para que un programa sea reubicable, es decir, que se pueda ejecutar en cualquier dirección de memoria.

Veamos a continuación cómo se soluciona esto en el 68000.

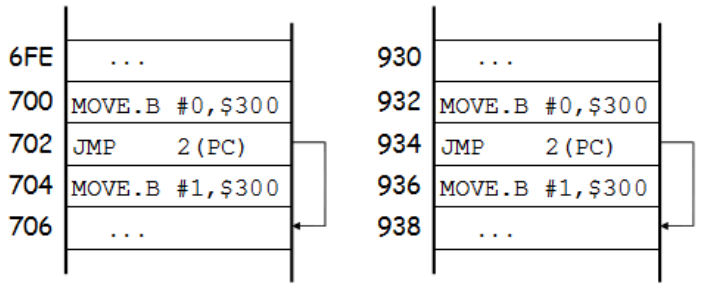
Modos de Direccionamiento ...Relativo al PC

Relativo al PC con Desplazamiento

- Se expresa como: d16 (PC)
- Idéntico al **d16 (An)** excepto que sólo puede utilizarse para operandos fuente (no alterables)

```
MOVE.B   30(PC), $700
MOVE.B   $700, 30(PC)  ¡ ILEGAL !
```

- Permite escribir programas reubicables



Aunque el problema de la dirección de salto también se podría solucionar con el direccionamiento indirecto por registro que hemos visto anteriormente, en el 68000 se suele utilizar una variante en la que el registro de indirección es el Contador de Programa (PC), en lugar de uno de los registros generales de direcciones.

Este **direccionamiento relativo al Contador de Programa** tiene dos variantes:

- Relativo al PC con desplazamiento

- Relativo al PC e indexado

El formato del **direccionamiento relativo al PC con desplazamiento** es idéntico al indirecto con desplazamiento, con la ya mencionada salvedad de que el registro de indirección es el Contador de Programa, y que este direccionamiento solamente puede utilizarse para hacer referencia a operandos fuente, es decir, que no puede utilizarse para modificar un operando.

Modos de Direccionamiento ...Relativo al PC

Relativo al PC e Indexado

- Se expresa como: d8 $(PC, X_i.z)$
- Idéntico al **d8(An, X_i.z)**
- Con la misma restricción que el Relativo a PC con Desplazamiento

El **direccionamiento relativo al PC e indexado**, también es equivalente al indirecto indexado, y con la misma restricción que el relativo al PC con desplazamiento.

Modos de Direccionamiento Direccionamientos

Modo	EA	REA	DEA	MEA	CEA	AEA	ADEA	AMEA	ACEA
Dn	✘	✘	✘			✘	✘		
An	✘	✘				✘			
(An)	✘		✘	✘	✘	✘	✘	✘	✘
(An)+	✘		✘	✘		✘	✘	✘	
-(An)	✘		✘	✘		✘	✘	✘	
d (An)	✘		✘	✘	✘	✘	✘	✘	✘
d (An, Xi)	✘		✘	✘	✘	✘	✘	✘	✘
Abs.W	✘		✘	✘	✘	✘	✘	✘	✘
Abs.L	✘		✘	✘	✘	✘	✘	✘	✘
d (PC)	✘		✘	✘	✘				
d (PC, Xi)	✘		✘	✘	✘				
Inmediato	✘		✘	✘					

No todos los modos de direccionamiento que hemos visto pueden utilizarse en cualquier situación y con cualquier instrucción. Para cada una de las instrucciones que vamos a ver se indican en su formato, mediante una nomenclatura, los modos de direccionamiento permitidos.

El cuadro adjunto muestra la nomenclatura utilizada para indicar los modos de direccionamiento permitidos en cada caso.

Como curiosidad, los nombres completos de los tipos de direccionamiento que se indican son los siguientes:

EA	*Efective Address*
REA	*Register Efective Address*
DEA	*Data Efective Address*
MEA	*Memory Efective Address*
CEA	*Control Efective Address*
AEA	*Control Efective Address*
ADEA	*Alterable Data Efective Address*
AMEA	*Alterable Memory Efective Address*
ACEA	*Alterable Control Efective Address*