3

MODOS DE DIRECCIONAMIENTO

3.1. Introducción

El campo de operación de una instrucción especifica la operación que se debe realizar. Ésta debe ser ejecutada sobre algunos datos almacenados en registros del computador o en palabras de memoria, es decir, sobre los operandos. El **modo de direccionamiento** especifica la *forma de interpretar la información contenida en cada campo de operando para localizar, en base a esta información, el operando*.

Los ordenadores utilizan técnicas de direccionamiento con los siguientes fines:

- **Dar versatilidad** de programación al usuario proporcionando facilidades tales como índices, direccionamientos indirectos, etc., esta versatilidad nos servirá para manejar estructuras de datos complejas como vectores, matrices, etc.
- **Reducir el número de bits del campo de operando**.

Al usuario que tiene poca experiencia, la variedad de modos de direccionamiento en un procesador le puede parecer excesivamente complicada. Sin embargo, la disponibilidad de diferentes esquemas de direccionamiento le da al programador experimentado flexibilidad para escribir programas que son más eficientes en cuanto a número de instrucciones y tiempo de ejecución.

Es tal la importancia de los modos de direccionamiento que la potencia de una máquina se mide tanto por su repertorio de instrucciones como por la variedad de modos de direccionamiento que es capaz de admitir.

Definición: Los **modos de direccionamiento** de un procesador son las *diferentes formas de transformación del campo de operando de la instrucción en la dirección del operando*.

En esta definición el término *dirección* debe interpretarse en su sentido más general de localización del operando, en cualquier lugar, y no en el sentido más estricto de dirección de memoria.

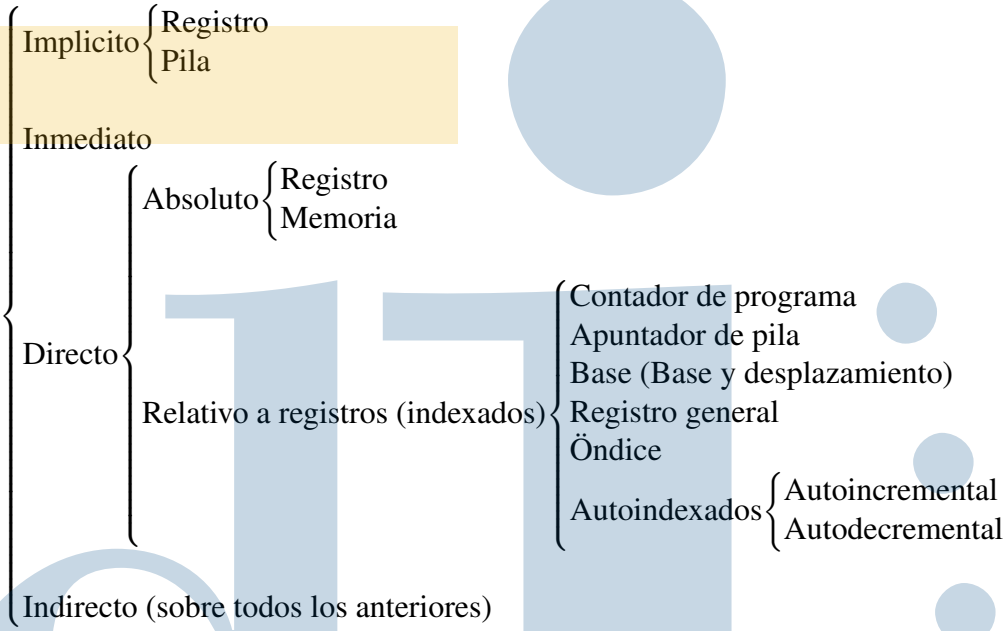

Fig. 3.1. Esquema-resumen de los modos de direccionamiento.

A la dirección obtenida de las transformaciones anteriores la llamaremos **dirección efectiva**. Esta dirección, en el caso de tratarse de una dirección de memoria, es la que se cargará en el MAR o registro de dirección de memoria.

Llamando x a la información del campo de operando y A a la dirección efectiva, la función f que a partir de x nos da A. constituirá el modo de direccionamiento empleado:

$$A = f(x)$$

En la evaluación de la función f pueden intervenir otras informaciones además de la información presente en el campo de operando de la instrucción. Estas informaciones pueden residir en registros del procesador o en memoria.

La especificación del modo de direccionamiento puede ir en el código de operación o en el campo de cada operando. Normalmente se codifica en el código de operación si el número de modos es pequeño, en caso contrario se codifica con cada operando, esta última forma de codificación favorece la ortogonalidad.

3.2. Modos de direccionamiento más usuales

En los párrafos siguientes se irán describiendo los modos de direccionamiento más frecuentes. La mayoría de estos modos son comunes a muchas máquinas, sin embargo, hay otros que sólo se usan en pocas máquinas. Un esquema general de los modos de direccionamiento más importantes puede verse en la figura 3.1.

Los modos de direccionamiento podrían clasificarse en **propios e impropios**: en los direccionamientos propios el operando está localizado en una dirección concreta de memoria, mientras que en los impropios en otros lugares tales como los registros del procesador, etc. Entre los modos de direccionamiento descritos en los párrafos siguientes, los tres primeros pueden considerarse impropios, mientras que los demás son direccionamientos propios.

3.2.1. Direccionamiento implícito

En este modo, llamado también **inherente**, *el operando se especifica en la misma definición de la instrucción*. El modo implícito se usa para hacer referencia a operandos de dos tipos:

- **Registros**: En el caso de que el código de operación se refiera en particular a un registro.
- **Operandos en la pila**: En el caso de que la operación se realice siempre sobre el dato situado en la cima de pila.

El primer caso es típico de las organizaciones de un solo acumulador. Generalmente en un ordenador de este tipo todas las instrucciones que actúan sobre el acumulador utilizan direccionamiento implícito.

En el segundo caso están la mayoría de las instrucciones de los ordenadores con organización de pila. Estas operaciones llevan implícitos los operandos que son los elementos de la cima de pila. Esto se debe a que en este tipo de máquinas la mayoría de las operaciones no tienen campos de dirección. También están en este caso las instrucciones PUSH y POP de la mayoría de los ordenadores cuyo operando implícito también es, como en el caso anterior, la cima de pila.

3.2.2. Direccionamiento inmediato (o literal)

En este modo es el operando el que figura en la instrucción no su dirección. En otras palabras *el campo de operando contiene él mismo, sin transformación alguna, la información sobre la que hay que operar*. Este modo es útil para inicializar registros o palabras de memoria con un valor constante.

3.2.3. Direccionamiento directo por registro

Se mencionó anteriormente que el campo de dirección de una instrucción puede especificar una palabra de memoria o un registro del procesador. Cuando se da este último caso se dice que el operando está especificado con **direccionamiento directo por registro**, en tal caso, *el operando reside en uno de los registros del procesador* que es seleccionado por un campo de registro de k bits en la instrucción. Este campo de k bits puede especificar uno de 2^k registros. Este modo es típico de los ordenadores con organización de registros de uso general.

Las **ventajas** de este modo son:

El acceso a los registros es muy rápido, por tanto el direccionamiento por registro debe usarse en las variables que se usen con más frecuencia para evitar accesos a memoria que son

más lentos, un ejemplo muy típico del uso de este direccionamiento son los **índices de los bucles**.

El número de bits necesarios para especificar un registro es mucho más pequeño que el necesario para especificar una dirección de memoria, esto es debido a que el número de registros del procesador es muy pequeño comparado con el número de direcciones de memoria. Sin embargo, hay que tener en cuenta que en los ordenadores modernos el número de registros ha aumentado considerablemente.

3.2.4. Direccionamiento directo (o absoluto)

Este es el modo de direccionamiento más sencillo. *El campo de dirección no necesita transformación alguna para dar la dirección efectiva*, es decir la función que transforma el campo de operando en la dirección efectiva es la identidad. Esto significa que el campo de operando es ya la dirección efectiva.

Este direccionamiento sólo se usa en ordenadores pequeños en que el programa siempre se sitúa en la misma zona de memoria ya que dificulta la **relocalización** de los programas, es decir que el código de los programas no dependa de su situación en memoria. En ordenadores más grandes, este modo está reservado para acceder a direcciones del sistema, que normalmente se refieren a operaciones de entrada y salida, ya que estas direcciones no dependen del programa.

3.2.5. Direccionamiento indirecto

En este modo *el campo de operando de la instrucción indica la localización de la dirección efectiva del operando*. El modo de direccionamiento indirecto puede adquirir diferentes formas según cuál sea el lugar donde se encuentre la dirección del operando. En general, todos los modos de direccionamiento tienen su versión indirecta que añade un eslabón más a la cadena del direccionamiento. Por ejemplo existe el direccionamiento indirecto por registro, en el que el registro especificado contiene la dirección del operando, no el operando mismo.

Este direccionamiento es útil cuando se trabaja con **apuntadores** ya que los apuntadores son variables que contienen las direcciones de los operandos, no los operandos mismos.

3.2.6. Direccionamiento relativo

Hay algunos modos de direccionamiento en que se hace uso de una propiedad muy generalizada de los programas denominada **localidad de referencia**, esta propiedad consiste en que *las direcciones referenciadas por los programas no suelen alejarse mucho unas de otras* y, por tanto, suelen estar concentradas en una parte de la memoria. Estas consideraciones nos llevan a la conclusión de que no es necesario utilizar todos los bits de la dirección de memoria en el campo de operando, basta utilizar los bits precisos para cubrir la parte de memoria donde estén incluidas las direcciones a las que el programa hace referencia. Esto puede hacerse tomando como referencia un punto de la memoria y tomando como campo de operando la diferencia entre ese punto y la dirección efectiva del operando. La dirección que se toma como punto de referencia puede residir en un registro de la CPU y, por tanto, sumando el contenido de ese

registro con el campo de operando obtendremos la dirección efectiva. Hay varios direccionamientos basados en esta técnica que reciben diferentes nombres dependiendo de cuál sea el registro en el que radica la dirección tomada como referencia. Todos ellos podrían catalogarse como direccionamientos **relativos a un registro**.

El direccionamiento denominado habitualmente relativo toma como valor de referencia el **registro contador de programa**. Cuando se usa este modo de direccionamiento, el campo de operando consiste en un número (normalmente con signo) que expresa la diferencia entre la dirección del dato y la dirección siguiente a la instrucción en curso (contenida en el contador de programa). Si el campo de operando, llamado en este caso **desplazamiento** u ***offset***, es positivo el operando residirá en una dirección posterior a la de la instrucción y si es negativo, en una dirección anterior.

Este modo de direccionamiento es usado muy frecuentemente en programas cuyo código deba ser independiente de la posición de memoria donde estén situados (programas relocalizables) ya que el desplazamiento es independiente de la localización del programa. También se usa con mucha frecuencia en instrucciones de bifurcación.

Los apartados siguientes se refieren a diferentes versiones de **direccionamientos relativos a registros**.

3.2.7. Direccionamiento por base y desplazamiento

Este modo de direccionamiento se fundamenta en la propiedad de localidad de referencia mencionada anteriormente. La dirección que se toma como referencia de la zona de memoria en la que están localizados los datos se deposita en un registro denominado **registro base** y el campo de operando indica la diferencia entre el registro base y la dirección del operando. Normalmente se toma como referencia (registro base) la dirección de comienzo de la zona de memoria ocupada por un programa. Por tanto, *la dirección efectiva del operando se calculará sumando el contenido del registro base con el campo de operando*.

Este modo de direccionamiento se usa en ordenadores que pueden mantener en memoria varios programas ya que, de esta forma, los diferentes registros base pueden contener las direcciones de comienzo de cada uno de los programas. Esto es muy útil porque facilita la relocalización de los programas: para situar el programa en una zona de memoria diferente bastará con cambiar el contenido de su registro base, no será necesario cambiar ninguno de los campos de operando.

3.2.8. Direccionamiento indexado

En este modo de direccionamiento, *la dirección del operando también se calcula sumando un registro de la CPU al campo de operando, este registro es un registro específico para este uso llamado* ***registro índice***. En los ordenadores con organización de registros generales, el registro índice puede ser cualquiera de los registros de la CPU. En los ordenadores en que el contador de programa es considerado como un registro de uso general (PDP-11 y VAX) el modo relativo es un caso particular del direccionamiento indexado. A la cantidad que hay que sumar al registro índice para conseguir la dirección del operando también se le llama **desplazamiento** u ***offset***. Este modo de direccionamiento es especialmente útil para el direccionamiento de vectores y

matrices en bucles ya que, si se quieren direccionar elementos consecutivos del vector o matriz, basta mantener en el desplazamiento la dirección del primer elemento e ir incrementando el registro índice. También sirve para acceder de forma relativa a elementos de vectores cercanos a uno dado, para ello, se carga la dirección del elemento de referencia en el registro índice y después se accede mediante direccionamiento indexado, con el desplazamiento adecuado, al anterior, al siguiente, etc., esto mismo también es aplicable a pilas, en que, algunas veces, hay que acceder a datos cercanos, por encima o por debajo, al dato señalado por algún apuntador. Una consecuencia de todo esto es una modalidad de direccionamiento indexado de que disponen algunos ordenadores, denominada **autoindexación**, que hace que el registro índice sea incrementado o decrementado en el tamaño del operando antes o después de acceder al mismo. Los ordenadores que poseen autoindexación incorporan los modos de direccionamiento descritos en los dos apartados siguientes. En algunos ordenadores existen variantes del direccionamiento indexado en que se obtiene la dirección del operando sumando el contenido de varios registros con el desplazamiento, esto puede servir para especificar el comienzo de un vector mediante un desplazamiento respecto a un registro y el elemento del vector mediante un registro índice.

3.2.9. Direccionamiento autoincremental o postincremental

En este modo, *la dirección del operando se encuentra en un registro y éste es incrementado, después de acceder al operando, en el tamaño del mismo*.

Este modo es útil para manejar vectores y matrices como se veía en el apartado anterior. También se puede utilizar para extraer datos de pilas (que crezcan hacia direcciones bajas) ya que, si el registro sobre el que se aplica este modo es el apuntador de pila, después de la operación el apuntador señalará al siguiente elemento de la pila.

3.2.10. Direccionamiento autodecremental o predecremental

En este modo *para obtener la dirección del operando hay que decrementar un registro en el tamaño del operando; el nuevo contenido del registro después de efectuar esa operación, es la dirección del operando*.

Este modo complementa al anterior y se emplea para direccionar elementos de vectores y matrices en orden descendente y también para introducir datos en las pilas ya que, si se aplica este modo sobre el apuntador de pila, conseguiremos que antes de efectuar el acceso el apuntador señale al siguiente hueco libre de la pila.

3.3. Visión general de los modos de direccionamiento

Un esquema del funcionamiento de los principales modos de direccionamiento se muestra en la figura 3.2. Por otra parte, en la tabla 3.1 se resumen las utilidades de los modos de direccionamiento usados con más frecuencia.

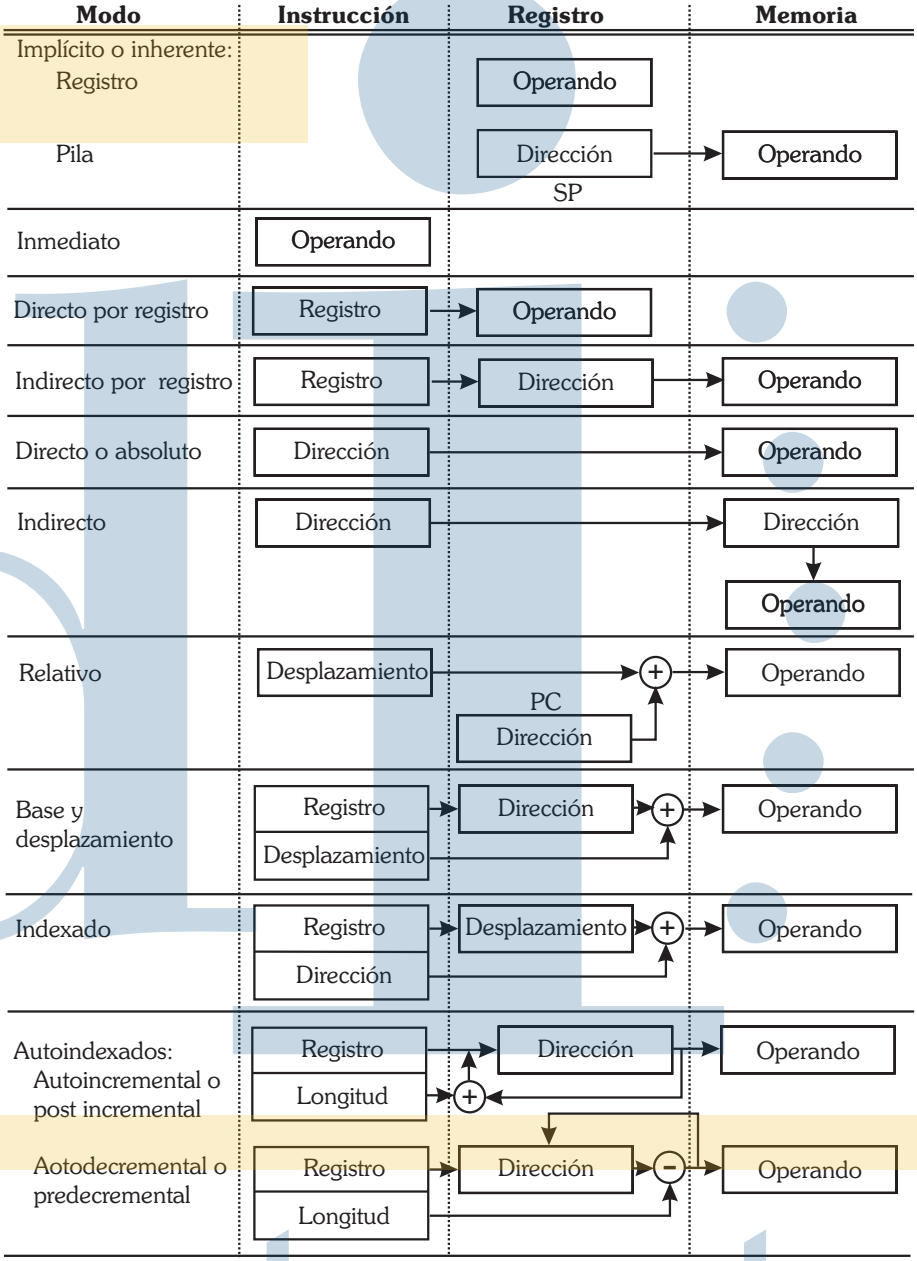

Fig. 3.2. Funcionamiento de los principales modos de direccionamiento.

3.4. Modos de direccionamiento en ordenadores reales

En este apartado estudiaremos los modos de direccionamiento que se usan en máquinas reales. Se verá que hay una gran parte de modos que son comunes a la mayoría de los procesadores y otros que son específicos de algunas máquinas.

Tabla 3.1. Utilidades de los modos de direccionamiento más importantes.

MODOS	UTILIDADES
Inmediato	Operaciones con constantes
Directo por registro	Variables locales de procedimientos no recursivos
Indirecto por registro	Variables referenciadas a través de apuntadores
Absoluto	Direcciones de sistema
Relativo	Variables globales
Indexado	Acceso a vectores, matrices y cadenas
Autoincremental	Desapilar parámetros de procedimientos Recorrido de vectores y cadenas
Autodecremental	Apilar parámetros de procedimientos Recorrido de vectores y cadenas hacia atrás

Nota:
En el capítulo siguiente se matizarán los conceptos de variables locales y globales. También se analizará la forma de almacenamiento de las variables locales de procedimientos recursivos.

3.4.1. PDP-11

Los modos de direccionamiento del PDP-11 (DEC, 1983) son bastante refinados y muy potentes. De los formatos de instrucción de esta máquina, vistos en el capítulo anterior, se desprende que cada operando se especifica mediante 6 bits, 3 que indican el modo de direccionamiento otros 3 que indican el registro sobre el que este modo actúa. El PDP-11 tendrá, por tanto, 8 modos de direccionamiento, aunque, como veremos, en la práctica tiene algunos más.

0: **Direccionamiento directo por registro** `R`n
El operando se encuentra en el registro Rn especificado.

1: **Direccionamiento indirecto por registro** `(R`n`)` o `@R`n
La dirección del operando se encuentra en el registro Rn.

2: **Direccionamiento autoincremental** `(R`n`)+`
La dirección del operando se encuentra en el registro Rn y éste se incrementa después del acceso en el tamaño del operando en bytes. Los incrementos posibles pueden ser dos: 1 si la operación se realiza sobre un byte o 2 si la operación se realiza sobre una palabra.

3: **Direccionamiento autoincremental indirecto** `@(R`n`)+`
El registro Rn especificado contiene un apuntador que señala la dirección del operando. El contenido del registro se incrementa después del acceso. Dado que, en este modo, el registro siempre apunta a la dirección del operando y las direcciones en el PDP-11 ocupan una palabra, el registro se incrementa en 2 cualquiera que sea el tamaño del operando. Este modo agrega una *indirección* al modo de direccionamiento anterior y se usa para trabajar con vectores de apuntadores.

4: **Direccionamiento autodecremental** `-(R`n`)`
El registro Rn se decrementa en la longitud del operando (1 si el operando es un byte o

2 si es una palabra). El contenido del registro después del decremento es la dirección del operando.

5: **Direccionamiento autodecremental indirecto** `@-(R`n`)`
El registro Rn se decrementa en 2 (que es la longitud de las direcciones en un PDP-11) y el resultado de esta operación es un apuntador que señala la dirección del operando. Este modo agrega una *indirección* al direccionamiento anterior y se usa para trabajar con vectores de apuntadores en sentido descendente.

6: **Direccionamiento indexado** X`(R`n`)`
La dirección del operando se obtiene sumando el desplazamiento (u offset) X al contenido del registro Rn (que no se modifica). El desplazamiento se codifica en la palabra siguiente a la instrucción.

7: **Direccionamiento indexado indirecto** `@`X`(R`n`)`
Sumando el contenido del registro Rn con el desplazamiento obtenemos la localización de la dirección del operando. El desplazamiento también se codifica en la palabra siguiente a la instrucción.

Modos de direccionamiento sobre el contador de programa:

En los direccionamientos anteriormente enumerados puede echarse en falta direccionamientos como el inmediato, directo, etc; estos direccionamientos no existen como tales en el código máquina del PDP-11, aunque, por supuesto, existen en el lenguaje ensamblador. Esto es así debido a que los modos citados se reducen a algunos de los anteriores cuando se aplican sobre el contador de programa (registro 7). Los modos aludidos son los siguientes:

Direccionamiento inmediato `#`*Constante*
Este modo es el autoincremental sobre el contador de programa debido a que el operando inmediato se localiza en la palabra siguiente a la instrucción. Este truco, que ahorra un código de direccionamiento, es posible debido a que el contador de programa es tratado como uno de los registros de uso general.

Direccionamiento absoluto `@#`*Dirección*
Este direccionamiento es el sustituto del direccionamiento directo: el operando se encuentra en la dirección especificada. Corresponde al direccionamiento indirecto autoincremental sobre el contador de programa. Este modo de direccionamiento se usa exclusivamente para direcciones que haya que especificar físicamente (direcciones de sistema) ya que si se utiliza para otros fines (por ejemplo, variables del programa) se provoca que el código sea dependiente de la posición y esto dificulta la relocalización de los programas ya que cada vez que el programa se instale en memoria en una dirección diferente habría que cambiar todas las variables que estuvieran direccionadas mediante direccionamiento absoluto.

Direccionamiento relativo *Dirección*
Es el sustituto del direccionamiento directo pero genera código independiente de la posición. En realidad se trata de direccionamiento indexado sobre el contador de programa. Este modo se utiliza para referirse a las direcciones (variables) del programa ya que lo

que queda registrado en el código es la diferencia entre la dirección de la variable y la dirección de la instrucción siguiente, por lo se que genera código independiente de la posición, y ello facilita la relocalización de los programas.

Direccionamiento indirecto relativo `@`*Dirección*
Este es el sustituto del direccionamiento indirecto convencional y es direccionamiento indirecto indexado sobre el contador de programa.

Como puede comprobarse, el hecho de que el contador de programa se considere un registro más ahorra codificaciones de modos de direccionamiento.

3.4.2. VAX

Los modos de direccionamiento del VAX (DEC, 1986, Baase, 1983) están claramente inspirados en los del PDP-11 aunque con algunas novedades bastante interesantes. De los formatos de instrucción, vistos en el capítulo anterior (apartado 2.6.2), se desprende que la especificación de cada operando se realiza mediante 8 bits, 4 que especifican el modo de direccionamiento y otros 4 que especifican el registro; por tanto la máquina tendrá 16 modos de direccionamiento, aunque, como veremos, hay modos que, según las circunstancias, pueden tener varias codificaciones diferentes y también, al igual que en el PDP-11, hay modos que, cuando se aplican sobre el contador de programa, reciben nombres diferentes y, en lenguaje ensamblador, tienen simbología distinta.

5: **Direccionamiento directo por registro** `R`n
El operando se encuentra en el registro Rn especificado si tiene un tamaño de 32 bits, si el tamaño del operando es inferior se toma la parte más baja. Si el operando tiene 64 bits (quadword) el operando lo forman los contenidos de los registros R_{n+1} y Rn tomados como uno solo.

6: **Direccionamiento indirecto por registro** `(R`n`)` o `@R`n
La dirección del operando se encuentra en el registro Rn.

7: **Direccionamiento autodecremental** `-(R`n`)`
El registro Rn se decrementa en la longitud del operando en bytes. Las longitudes posibles de los operandos a estos efectos van de 1 (byte) a 16 (*octaword* o números en punto flotante). El contenido del registro después de la operación es la dirección del operando.

8: **Direccionamiento autoincremental** `(R`n`)+`
La dirección del operando se encuentra en el registro y éste se incrementa después del acceso en el tamaño del operando en bytes (de 1 a 16).

9: **Direccionamiento autoincremental indirecto** `@(R`n`)+`
El registro contiene un apuntador que señala a la dirección del operando. El contenido del registro se incrementa después del acceso. Dado que, en este modo, el registro siempre apunta a la dirección del operando, y ésta, en el VAX, ocupa siempre una doble palabra independientemente de la longitud del dato al que apunte, el registro se incrementa en 4 cualquiera que sea el tamaño del operando. Este modo agrega una textitindirección al direccionamiento anterior y se usa para trabajar con vectores de apuntadores.

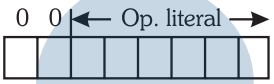

Fig. 3.3. Formato de un operando en modo literal en el VAX.

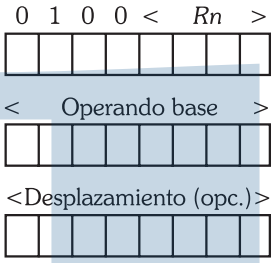

Fig. 3.4. Formato de un operando en modo indexado en el VAX.

A,C,E: **Direccionamiento por desplazamiento** X`(R`n`)`

La dirección del operando se obtiene sumando el desplazamiento X al contenido del registro Rn (que no se modifica). La suma se realiza extendiendo el signo del desplazamiento a 32 bits si su tamaño es menor. El desplazamiento se codifica tras la especificación de operando. Dependiendo del tamaño del desplazamiento, la codificación es distinta (A: byte `B^`X`(R`n`)`, C: palabra `W^`X`(R`n`)`, E: doble palabra `L^`X`(R`n`)`. En lenguaje ensamblador, no es necesario especificar el tamaño del desplazamiento a menos que se quiera forzarlo; normalmente, el ensamblador elige el tamaño adecuado dependiendo del valor del desplazamiento. Este modo es similar al direccionamiento indexado del PDP-11.

B,D,F: **Direccionamiento indirecto por desplazamiento** `@`X`(R`n`)`

Sumando el desplazamiento X al contenido del registro obtenemos la localización de la dirección del operando. El desplazamiento también se codifica después de la especificación de operando y dependiendo de su tamaño su codificación también es diferente (B: byte `@B^`X`(R`n`)`, D: palabra `@W^`X`(R`n`)`, F: doble palabra `@L^`X`(R`n`)`). De igual forma que en el modo anterior en ensamblador no es necesario especificar el tamaño del desplazamiento ya que el ensamblador toma el tamaño más adecuado dependiendo del valor del desplazamiento.

0-3: **Literal** `S^#`$Literal$ (S: Short inmediate)

El direccionamiento literal corresponde a los valores de la codificación de direccionamiento cuyos dos primeros bits son 0. Este modo rompe un poco la ortogonalidad de la máquina, ya que el resto de los modos necesitan trabajar con algún registro. Dos de los bits de codificación de modo se invaden por el operando literal que ocupa 6 bits (los dos invadidos más los cuatro correspondientes al campo de registro). Este modo sirve para codificar pequeñas constantes positivas (de 0 a 63). En la figura 3.3 se muestra un esquema de la codificación del direccionamiento literal en código máquina.

4: **Indexado** b`[R`n`]`

Este modo es diferente al direccionamiento indexado del PDP-11 ya que su propósito

es mucho más general. En el PDP-11 el desplazamiento sólo puede ser una constante, en el VAX puede ser cualquier tipo de operando (b). Para obtener la dirección efectiva del operando, se suma el contenido del registro índice (Rn) multiplicado por el tamaño del operando (en bytes) con la **dirección** del operando base (b) que puede expresarse en diferentes modos (llamados modos indexables, del 6 al F, modos en que el operando reside en memoria, a excepción del propio direccionamiento indexado). El registro índice no puede ser el contador de programa ($R15$). En la figura 3.4 puede verse la codificación máquina del modo indexado; en ella, el desplazamiento se refiere al operando base y puede tener de 0 a 4 bytes.

Modos de direccionamiento sobre el contador de programa:

Dado que, al igual que en el PDP-11, el contador de programa es considerado como un registro más (R15), puede también utilizarse con algunos de los modos anteriores. Esto da lugar a los direccionamientos siguientes:

Direccionamiento inmediato `#`*Constante*

Este modo de direccionaiento es el autoincremental sobre el contador de programa ya que el operando inmediato se localiza después de la especificación del operando.

Direccionamiento absoluto `@#`*Dirección*

Este direccionamiento es el sustituto del direccionamiento directo: el operando se encuentra en la dirección especificada, corresponde al direccionamiento indirecto autoincremental sobre el contador de programa. Este direccionamiento se usa exclusivamente para direcciones que haya que especificar físicamente (direcciones de sistema) ya que si se usa para otros fines este direccionamiento genera código dependiente de la posición.

Direccionamiento relativo *Dirección*

Es el sustituto del direccionamiento directo pero genera código independiente de la posición. En realidad se trata de direccionamiento por desplazamiento sobre el contador de programa. Este modo se utiliza para referirse a las direcciones del programa.

Direccionamiento indirecto relativo `@`*Dirección*

Éste es el sustituto del direccionamiento indirecto convencional y, realmente, es direccionamiento indirecto por desplazamiento sobre el contador de programa. Tanto en este modo como el anterior, el tamaño del desplazamiento lo decide el ensamblador al realizar la traducción.

3.4.3. MC68000 y derivados

En el MC68000 (Harman, 1989) la especificación de operando se realiza mediante 6 bits: 3 para modo y 3 para registro. A diferencia del PDP-11, el PC no es uno de los registros de uso general, por tanto, el artificio que allí se usaba para codificar algunos modos de direccionamiento, aquí no será válido. Para codificar los direccionamientos que no precisan registro, esta máquina utiliza extensión en los códigos de manera que, si el código de modo es $111_{(2}$, el campo de registro es invadido por el código por lo que así se pueden codificar hasta 8 modos más. Por otra parte, el mismo direccionamiento lleva implícito el tipo de registro sobre el que trabaja (direcciones o datos).

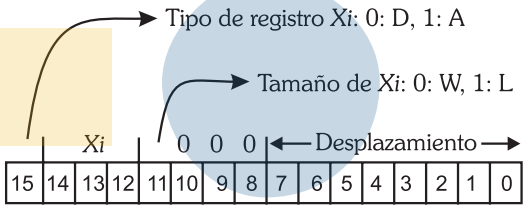

Fig. 3.5. Formato de la palabra adicional del modo indexado completo del MC68000.

0: **Direccionamiento directo por registro de datos** `D`n
El operando se encuentra en el registro de datos especificado.

1: **Direccionamiento directo por registro de direcciones** `A`n
El operando se encuentra en el registro de direcciones especificado.

2: **Direccionamiento indirecto por registro de direcciones** `(A`n`)`
La dirección del operando se encuentra en el registro de direcciones.

3: **Direccionamiento postincremental** `(A`n`)+`
La dirección del operando se encuentra en el registro de direcciones especificado y éste se incrementa después del acceso. Después del acceso el registro se incrementa en la longitud del operando en bytes (1, 2 o 4).

4: **Direccionamiento predecremental** `-(A`n`)`
El contenido del registro de direcciones se decrementa en la longitud del operando (1, 2 o 4). El resultado es la dirección del operando.

5: **Direccionamiento indexado simple** d_{16}`(A`n`)`
La dirección efectiva del operando se obtiene sumando el desplazamiento (u *offset*) d_{16} al contenido del registro de direcciones especificado (que no se modifica). El desplazamiento se codifica en la palabra siguiente a la instrucción.

6: **Direccionamiento indexado completo** d_8`(A`n`,`Xi`)`
Para obtener la dirección efectiva del operando hay que sumar el contenido del registro de direcciones An con el registro índice Xi, que puede ser tanto de datos como de direcciones, y con el desplazamiento d_8 tomado en complemento a 2 y extendiendo su signo. Este modo tiene dos versiones según se tomen los 16 últimos bits del registro índice Xi, con signo extendido (en ensamblador se indica d_8`(A`n`,`Xi`.W)` o simplemente d_8`(A`n`,`Xi`)` o los 32 (en cuyo caso de indica d_8`(A`n`,`Xi`.L)`) La codificación máquina de este modo exige una palabra adicional, denominada **palabra de extensión** que tiene el formato mostrado en la figura 3.5.

En el 68020 (y siguientes) este modo tiene diversas variantes cuya palabra de extensión se codifica según uno de los formatos de la figura 3.6. La codificación de esas variantes reside en los últimos bits de esa misma palabra de extensión:

Indexado con factor de escala y desplazamiento corto (d_8`,A`n`,`$Xi.S * Escala$`)`
El cálculo de la dirección efectiva se efectúa de la siguiente forma:

$$A = d_8 + An + Xi.S * Escala$$

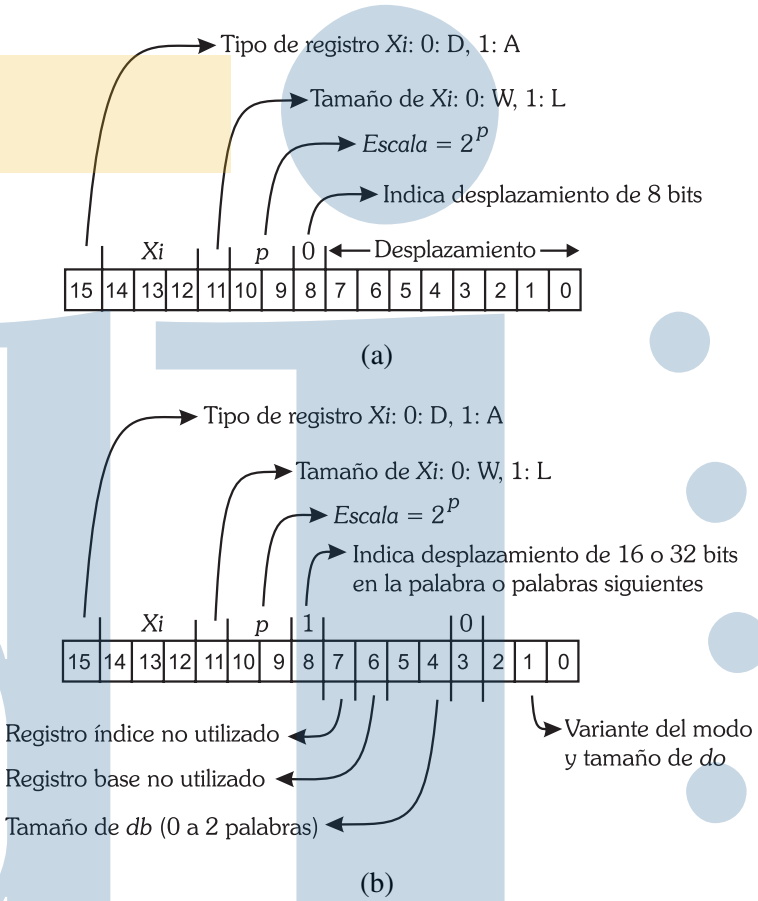

Fig. 3.6. Formato de la palabra adicional de las variantes del modo indexado completo del MC68020.

donde S = W o L y $Escala$ = 1, 2, 4 u 8.

(Se utiliza el formato de la figura 3.6 (a)) Debe observarse que este direccionamiento es compatible con el indexado completo del MC68000 haciendo $Escala$ = 1.

Indexado por factor de escala y desplazamiento largo $(db, \texttt{An}, Xi.S{*}Escala)$ En este modo el cálculo de la dirección efectiva se realiza así:

$$A = db + An + Xi.S * Escala$$

La diferencia entre este direccionamiento y el anterior radica en que el desplazamiento db (llamado desplazamiento base) puede tener 16 o 32 bits y que los tres parámetros son opcionales, por lo que este modo puede adoptar múltiples formas como, por ejemplo, direccionamiento indirecto por registro de datos. Se utiliza el formato de la figura 3.6(b) codificándose el desplazamiento db en la(s) palabra(s) siguiente(s).

Indirecto postindexado $([db, \texttt{An}], Xi.S * Escala, do)$ La dirección efectiva en este modo se calcula de la siguiente forma:

$$A = [db + An] + Xi.S * Escala + do$$

En esta expresión los corchetes representan un acceso a memoria adicional, es decir, una indirección más; por otra parte, do (desplazamiento exterior) representa un

desplazamiento de 16 o 32 bits que se codifica en la(s) palabra(s) siguientes al resto de las informaciones. Todos los parámetros son opcionales. Se emplea el formato de la figura 3.6 (b) teniendo en cuenta que los desplazamientos db y do se codifican en las palabras siguientes.

Indirecto preindexado `([`db`,An,`$Xi.S{*}Escala$`],`do`)` La dirección efectiva en este modo se calcula de la siguiente forma:

$$A = [db + An + Xi.S * Escala] + do$$

Este modo difiere del anterior en el momento de sumar el registro índice: en este caso la suma se realiza antes de la indirección. Al igual que en los modos anteriores todos los parámetros son opcionales.

Los siguientes modos de direccionamiento corresponden a la extensión del código de modo al campo de registro cuando el código principal de modo es $111_{(2}$, es decir 7_{10}. Evidentemente, estos modos no actúan sobre ningún registro de uso general.

7.0: **Direccionamiento absoluto corto** $Dirección_{16}$
Se especifica directamente la dirección del operando que se extiende en signo hasta los 32 bits. La dirección del operando está en la palabra siguiente a la instrucción. Este modo y el siguiente son equivalentes al modo directo de otras máquinas.

7.1: **Direccionamiento absoluto largo** $Dirección_{32}$
Se especifica la dirección del operando de la que se toman los 24 bits menos significativos (en el MC68000, en el MC68020 y siguientes esto no es necesario). La dirección reside en las dos siguientes palabras a la instrucción. La diferencia entre este direccionamiento y el anterior radica en que con el modo absoluto largo se puede acceder a todo el espacio direccionable de la máquina mientras que con el corto, sólo se pueden direccionar 32 K al principio de (bit 15=0) la memoria y otros 32 K al final (bit 15=1).

7.2: **Direccionamiento relativo simple** $d_{16}(PC)$ o $Etiqueta$
La dirección del operando se calcula sumando un desplazamiento de 16 bits con el contador de programa. Es equivalente al direccionamiento indexado simple pero actuando sobre el contador de programa.

7.3: **Direccionamiento relativo completo** $d_8(PC, Xi)$
Para calcular la dirección del operando hay que sumar los contenidos del contador de programa y del registro índice con el desplazamiento de 8 bits. Es equivalente al direccionamiento indexado completo sobre el contador de programa y la codificación de la palabra de extensión es idéntica.

En el procesador MC68020 este modo de direccionamiento tiene las mismas variantes que el direccionamiento indexado completo y utiliza los mismos formatos en la palabra de extensión (figura 3.6); la única diferencia es que se aplica sobre el contador de programa.

7.4: **Direccionamiento inmediato** *#Expresión*
En este modo el operando se encuentra en la (o las) palabra siguiente a la instrucción. Si el tamaño del operando es de 8 bits, éste se encuentra en los 8 bits de menor peso de la palabra siguiente a la instrucción (los otros 8 son 0).

Fig. 3.7. Formato general de operandos del 8086.

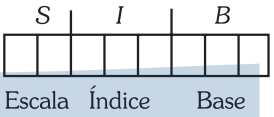

Fig. 3.8. Formato del byte SIB.

3.4.4. i-8086 y derivados

Los modos de direccionamiento del 8086 (Crawford & Gelsinger, 1987) son muy irregulares. Recordando la codificación de los operandos de esta máquina, mostrada en la figura 3.7, vemos que hay un campo para un registro (reg), que especifica uno de los operandos, y otros dos campos (mod y r/m) para el otro. En la tabla 3.2 se muestran todas las combinaciones posibles de los valores de estos campos. Los modos de direccionamiento de esta máquina no son en absoluto ortogonales, se quiere decir con esto que no se pueden aplicar todos los modos sobre todos los registros. Por otra parte, no existe el modo inmediato sino códigos de operación específicos para permitir el direccionamiento inmediato. Los modos autoindexados no están soportados por esta máquina. Como se desprende de la tabla, la mayoría de los modos son indirectos por registro, indexados o directos por registro; en este caso ($mod = 11$), existen dos posibilidades en función del tamaño del operando.

El 80386, en modo real es completamente compatible con el 8086, pero cuando trabaja en modo protegido, es decir, con segmentos de 32 bits, es mucho más regular en la codificación de los direccionamientos (ver tabla 3.3). Existe un modo nuevo que requiere un byte adicional denominado SIB (escala, índice, base) que se añade al byte de operandos y cuyo formato se muestra en la figura 3.8. Este byte adicional especifica un factor de escala y dos registros (base e índice). Para calcular la dirección con este modo se multiplica el contenido del registro índice

Tabla 3.2. Modos de direccionamiento del i-8086

	mod			
r/m	00	01	10	11
000	(BX+SI)	(BX+SI+d_8)	(BX+SI+d_{16})	AX o AL
001	(BX+DI)	(BX+DI+d_8)	(BX+DI+d_{16})	CX o CL
010	(BP+SI)	(BP+SI+d_8)	(BP+SI+d_{16})	DX o DL
011	(BP+DI)	(BP+DI+d_8)	(BP+DI+d_{16})	BX o BL
100	(SI)	(SI+d_8)	(SI+d_{16})	SP o AH
101	(DI)	(DI+d_8)	(DI+d_{16})	BP o CH
110	Directo	(BP+d_8)	(BP+d_{16})	SI o DH
111	(BX)	(BX+d_8)	(BX+d_{16})	DI o BH

Tabla 3.3. Modos de direccionamiento del i-80386

r/m \ mod	00	01	10	11
000	(EAX)	(EAX+d_8)	(EAX+d_{32})	EAX o AL
001	(ECX)	(ECX+d_8)	(ECX+d_{32})	ECX o CL
010	(EDX)	(EDX+d_8)	(EDX+d_{32})	EDX o DL
011	(EBX)	(EBX+d_8)	(EBX+d_{32})	EBX o BL
100	SIB	SIB y d_8	SIB y d_{32}	ESP o AH
101	Directo	(EBP+d_8)	(EBP+d_{32})	EBP o CH
110	(ESI)	(ESI+d_8)	(ESI+d_{32})	ESI o DH
111	(EDI)	(EDI+d_8)	(EDI+d_{32})	EDI o BH

por el factor de escala (que puede ser 1, 2, 4 u 8), se le suma el registro base y en algunas ocasiones (ver tabla 3.3) un desplazamiento de 8 o 32 bits. El modo SIB es útil para direccionar elementos de vectores de longitudes diferentes en bucles. Es una alternativa a los modos autoindexados que esta máquina no soporta.

3.4.5. Arquitectura SPARC

La arquitectura SPARC tiene un número muy limitado de modos de direccionamiento por ser una arquitectura RISC. Básicamente los direccionamientos usados son los siguientes:

Direccionamiento inmediato *Constante*
Como es sabido, en este modo el operando es una constante que se especifica en la misma instrucción. La característica más interesante de este modo en la arquitectura SPARC es que no se añade información alguna a la instrucción para codificar la constante. La constante tiene reservado su espacio en la misma instrucción.

Direccionamiento directo por registro %rn
Es el modo de direccionamiento más usado en esta máquina, ya que se trata de una máquina registro-registro. El operando reside en el registro especificado por un número de 5 bits.

Direccionamiento indexado [%rn + %rm] o [%$rn + d$]
Este modo es el utilizado para los accesos a memoria y tiene dos variantes, en función de la forma de calcular la dirección eficaz del operando:

- La dirección del operando se calcula sumando los contenidos de dos registros ([%rn+ %rm]).
- La dirección del operando se calcula sumando el contenido de un registro y un desplazamiento ([%rn+d]). Un caso particular de este modo es el direccionamiento indirecto por registro, en que el desplazamiento es 0 ([%rn]).

Bibliografía y referencias

BAASE, S. 1983. *VAX-11 Assembly Language Programming*. Prentice-Hall.

CRAWFORD, J, & GELSINGER, P. 1987. *Programming the 80386 (Featuring 80386/80387)*. Sybex. Existe traducción al castellano: Programación del 80386/387, Anaya Multimedia, 1991.

DE BLASI, M. 1990. *Computer Architecture*. Addison Wesley.

DEC. 1983. *PDP-11 Architecture Handbook*. Digital Equipment Corporation.

DEC. 1986. *VAX Architecture Handbook*. Digital Equipment Corporation.

HARMAN, T.L. 1989. *The Motorola MC68020 and MC68030 Microprocessors, Assembly Language, Interfacing and Design*. Vol. II. Prentice-Hall International.

TANENBAUM, A.S. 2006. *Structured Computer Organization*. 5 edn. Prentice-Hall International. Existe traducción al castellano de la edición anterior: Organización de computadores: un enfoque estructurado, 4ª edición, Prentice-Hall Hispanoamericana, 2000.

CUESTIONES Y PROBLEMAS

Nota importante: Para la resolución de muchos problemas de este capítulo y del siguiente es necesario consultar los apéndices.

3.1 Traducir a código máquina las siguientes instrucciones de lenguaje ensamblador del PDP-11:

a) `ADD R2,R3` b) `INC (R5)+` c) `CLR -(SP)` d) `ADD (R2),R1`
e) `CLR 5(R3)` f) `MOV #4,R1` g) `INCB A` h) `MOV #A, A`

Se puede suponer, cuando sea necesario, que cada instrucción está localizada en la dirección $077420_{(8}$ y que el símbolo `A` representa a la dirección $063246_{(8}$.

3.2 Poner cada una de las siguientes instrucciones del VAX en código máquina:

a) `INCW (R5)+` b) `MOVL (R10)+, -(SP)`
c) `BICL3 #18, (R4)+, -(R6)` d) `MOVB #4, (R7)`
e) `ADDL3 R8, R9, 5(R4)[R3]` f) `SUBW2 4(R6), 2500(R7)[R4]`

3.3 Traducir las siguientes instrucciones de VAX a código máquina, suponiendo que el símbolo `A` representa a la dirección 0034AAF8H, que el símbolo `B` representa a la constante $258_{(10}$ y que cada instrucción se encuentra en la dirección 0034AB70H (las instrucciones son independientes unas de otras):

a) `CLRW A` b) `MOVL #B, A`
c) `MOVB 4(R1), A` d) `SUBW2 A, 5(R2)[R4]`
e) `ADDL3 (R5)+, #4, A` f) `MULL3 #B, A, A[R3]`

3.4 Describir los efectos de cada una de las instrucciones de los problemas 3.1, 3.2 y 3.3.

3.5 Calcular la localización de los operandos de las siguientes instrucciones correspondientes a la arquitectura SPARC (si es necesario, suponer ciertos valores de los contenidos de los registros):

a) `or %l0, %g1, %o1` b) `stb %o0, [%i0+%l1]`
c) `ldub [%i0 + 40], %o0` d) `subcc %o0, 10, %g0`

3.6 Traducir a lenguaje máquina las instrucciones del problema anterior

3.7 Sean las siguientes instrucciones correspondientes a un VAX (las instrucciones son independientes unas de otras):

```
MOVL (R3)+, (R3)+
SUBW2 #2, 4(R3)
ADDW2 X, X[R4]
```

a) Calcular la dirección de todos los operandos para cada una de esas instrucciones. Suponer que los registros $R3$ y $R4$ contienen, respectivamente, los valores AB1209F0H y 5 y que el símbolo `X` representa a la dirección de memoria 00AB02F0H.

b) Escribir cada una de las instrucciones en lenguaje máquina optimizando el espacio de memoria ocupado. Suponer, además de lo anterior, que cada instrucción se encuentra a partir de la dirección 00AB2E0AH.

3.8 Suponiendo que el registro $A3$ de un MC68000 contiene 00041E0AH y que el contenido de las palabras de memoria situadas a partir de la dirección 41E08H es 4F90H, A56AH, 60C8H y 678BH, determinar cuáles serán los contenidos finales de los registros $A3$ y $D2$ después de la ejecución de cada una de las siguientes instrucciones:

a) `MOVE.W A3, D2` b) `MOVE.W (A3)+, D2`
c) `MOVE.W (A3), D2` d) `MOVE.W #$100F, D2`
e) `MOVE.W -(A3), D2` f) `MOVE.W 4(A3), D2`

Nota: En el ensamblador del MC68000, el prefijo $ indica que la constante siguiente debe interpretarse en hexadecimal y la última letra del nemónico y separada por un punto de éste, indica el tamaño de los operandos.

3.9 Analizar los efectos de las siguientes instrucciones correspondientes a un MC68020, suponiendo que los registros $A5$, $D2$ y $D3$ contienen respectivamente los valores 08F9A204H, A0BAC902H y FAB9A2F0H. Suponer valores arbitrarios en las direcciones de memoria afectadas:

a) `ADDQ.L #1, D3` b) `SUBQ.W #5, D3`
c) `AND.B 8(A5), D3` d) `EOR.L D2, 4(A5,D3)`
e) `LEA 8(A5,D3.L), A2` f) `CLR.W 4(A5,D3*4)`
g) `NEG.L -(A5)` h) `ADDI.W #400, (A5)+`

3.10 ¿Cuáles de las instrucciones del problema anterior no se podrían ejecutar en un MC68000?

3.11 Traducir a código máquina cada una de las instrucciones de los problemas 3.8 y 3.9.

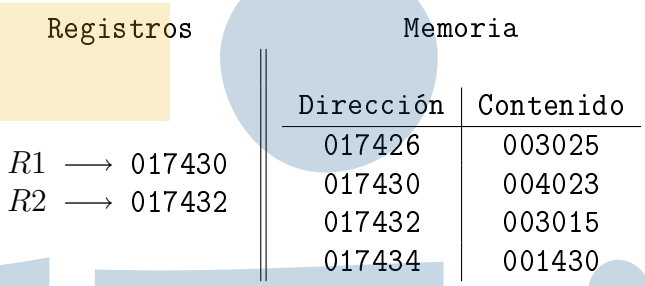

Fig. 3.9.

3.12 Estudiar la compatibilidad de los direccionamientos de las siguientes instrucciones correspondientes a un MC68000:

a) `ADD.B D2, (A3)+` b) `ADDI.W #5, A3` c) `CLR.L (A3)+`
d) `ASL.L #4, (A2)` e) `LEA D2, A3` f) `ROR.W D3,D4`

3.13 Sea el fragmento siguiente de programa de un PDP-11:

```
MOV #A, R0
ADD #4, (R0)+
```

Suponiendo que la dirección de memoria representada por `A` es la $065702_{(8}$ y que, antes de la ejecución de esas instrucciones, contiene $010234_{(8}$:

a) ¿Qué contiene `A` después? ¿Qué contiene $R0$?

b) Codificar el fragmento de programa en lenguaje máquina.

3.14 Supongamos un PDP-11 cuya situación es la mostrada en la figura 3.9. ¿Qué habrá cambiado después de ejecutarse la instrucción

```
MOV R1, (R2)+?
```

3.15 Supongamos que los registros $R1$ y $R2$ de un PDP-11 contienen $042753_{(8}$ y $142752_{(8}$. ¿Cómo quedarán estos registros después de las siguientes instrucciones?

a) `SUB R1, R2` b) `SUB R2, R1` c) `INC R1` d) `INCB R1`
e) `CLR R1` f) `ADD R2, R1` g) `ADD R1, R2` h) `DEC R1`

3.16 Escribir en código máquina la instrucción

```
SUBB2 A, A
```

del VAX suponiendo que reside a partir de la dirección 0A0D02D9H y que la etiqueta `A` representa a la dirección 0A0D02C0H.

```
COMIENZO:   .......
            .......
            MOV #CADENA1, R4
            MOV #CADENA2, R5
            MOV N, R1
LOOP:       CMPB (R4)+, (R5)+
            BNE FIN
            SOB R1, LOOP
            MOV #177777, R0
            BR COMIENZO
FIN:        CLR R0
            BR COMIENZO
```

Fig. 3.10.

3.17 a) Codificar el fragmento de programa mostrado en la figura 3.10 correspondiente a un PDP-11, en lenguaje máquina, suponiendo que las etiquetas `CADENA1`, `CADENA2`, `N`, `COMIENZO` y `LOOP` representan las direcciones $000500_{(8}$, $001000_{(8}$, $001500_{(8}$, $002600_{(8}$ y $002700_{(8}$ respectivamente.

b) Explicar cuál puede ser el propósito del citado fragmento de programa.

3.18 Un PDP-11 tiene su memoria en la situación que se muestra en la figura 3.11 (octal).

a) Decodificar el programa existente a partir de la dirección $004352_{(8}$.

b) ¿Cuál puede ser el propósito de ese programa?

3.19 Supongamos que un VAX se encuentra en el estado mostrado en la figura 3.12. Si el símbolo `A` representa a la dirección 00404F0AH y la instrucción

```
ADDW3 R7, (R6)+, A
```

se encuentra en la dirección 00405006H:

a) ¿Qué cambios provocará esta instrucción?

b) Codificar esa instrucción en código máquina.

3.20 Las versiones A y B de la figura 3.13 corresponden al código de un mismo programa en ensamblador de PDP-11. Si las etiquetas `X`, `N`, `M`, `R` y `C` representan, respectivamente, a las direcciones $007720_{(8}$, $066040_{(8}$, $066050_{(8}$, $066060_{(8}$ y $076400_{(8}$, y que las direcciones `N` y `M`, respectivamente, contienen, los valores n y $2(n-1)$:

a) Escribir ambos programas en lenguaje máquina.

b) ¿Realizan ambos programas la misma tarea? Si no es así, modificarlos para que realicen la misma función.

c) ¿Qué tarea efectúan?

d) ¿Cuál de las dos versiones es más rápida? ¿Por qué?

Dirección	Contenido
004352	012703
004354	003000
004356	012704
004360	003500
004362	016701
004364	176356
004366	121327
004370	000060
004372	002413
004374	121327
004376	000071
004400	003010
004402	112367
004404	176334
004406	162767
004410	000060
004412	176326
004414	016724
004416	176322
004420	000401
004422	005203
004424	077120
004426	000000

Fig. 3.11.

Registros

$R6 \longrightarrow$ 00404F08
$R7 \longrightarrow$ A034378A

Memoria

Dirección	Contenido
404F08	A4
404F09	C2
404F0A	FB
404F0B	77

Fig. 3.12.

```
        Versión A                Versión B

        CLR R0                   CLR R0
        MOV M, R1                MOV #X, R1
C:      ADD X(R1), R0            MOV N, R2
        SUB #2, R1       C:      ADD (R1)+, R0
        BGE C                    SOB R2, C
        MOV R0, R                MOV R0, R
```

Fig. 3.13.

Dirección	Contenido
010406	005003
010410	012701
010412	004620
010414	016704
010416	177674
010420	010321
010422	005203
010424	077403
010426	000000

A

Dirección	Contenido
010406	005003
010410	016704
010412	177700
010414	010363
010416	004620
010420	005203
010422	077404
010424	000000
010426	000000

B

Fig. 3.14.

3.21 Las partes A y B de la figura 3.14 corresponden a la misma zona de memoria de un PDP-11 que incluye el código del mismo programa en dos versiones:

a) Escribir ambas versiones en lenguaje ensamblador.

b) ¿Qué efectos tiene cada versión?

c) Si ambas versiones tuvieran efectos diferentes realizar las modificaciones necesarias para que ambas realicen la misma función.

d) ¿Cuál de las dos versiones es más rápida?

3.22 Sea un PDP-11 en cuya memoria reside el programa de la figura 3.15:

a) Traducir el programa a código máquina, suponiendo que el símbolo `A` representa a la dirección $004574_{(8}$ y el símbolo `D` a la $077642_{(8}$.

b) Explicar los cambios que produce este programa en los registros y la memoria, en este caso, y en el caso de que las cadenas `A` y `B` contengan otros caracteres.

```
A:      .ASCII /ESTO ES UNA CADENA/
L1=.-A
B:      .ASCII /ESTO ES OTRA CADENA/
L2=.-B
C:      .WORD 0
        .......
        .......
D:      CMP #L1,#L2
        BLT E
        MOV #L2,R3
        BR F
E:      MOV #L1,R3
F:      MOV #A,R0
        MOV #B,R1
        MOV R0,R2
G:      CMPB (R0)+, (R1)+
        BEQ H
        SUB R2,R0
        MOV R0,C
        HALT
H:      SOB R3,G
        CLR C
        HALT
```

Fig. 3.15.